Benutzungshandbuch

Scratch-Kara-Simulator

Version 1.1 (25.01.2011)

Dietrich Boles

Universität Oldenburg

Inhaltsverzeichnis

1 Einleitung

Programmieranfänger haben häufig Schwierigkeiten damit, dass sie beim Programmieren ihre normale Gedankenwelt verlassen und in eher technisch-orientierten Kategorien denken müssen, die ihnen von den Programmiersprachen vorgegeben werden. Gerade am Anfang strömen oft so viele inhaltliche und methodische Neuigkeiten auf sie ein, dass sie das Wesentliche der Programmierung, nämlich das Lösen von Problemen, aus den Augen verlieren.

1.1 Das Kara-Modell

Das Kara-Modell ist mit dem Ziel entwickelt worden, dieses Problem zu lösen. Mit dem Kara-Modell wird Programmieranfängern ein einfaches, aber mächtiges Modell zur Verfügung gestellt, mit dessen Hilfe Grundkonzepte der (imperativen) Programmierung auf spielerische Art und Weise erlernt werden können. Programmierer entwickeln so genannte „Kara-Programme“, mit denen sie einen virtuellen Marienkäfer namens Kara durch eine virtuelle Landschaft steuern und bestimmte Aufgaben lösen lassen.

Das Kara-Modell wurde an der ETH Zürich entwickelt und ist nun in die schweizerische Bildungsplattform SwissEduc integriert. Viele Informationen rund um das Modell stehen auf der Kara-Website zur Verfügung:

http://www.swisseduc.ch/informatik/karatojava/

1.2 Das Scratch-Kara-Modell

Eine Programmiersprache, in der man Kara-Programme entwickeln kann, ist bspw. Java. Java ist eine textuelle Programmiersprache, bei der Programme vor ihrer Ausführung zunächst von einem Compiler auf syntaktische Korrektheit überprüft werden müssen. Gerade Programmieranfänger haben aber häufig Probleme mit der beim Eingeben des Programmcodes erforderlichen Genauigkeit und den Fehlermeldungen des Compilers. Diesbezüglich versprechen visuelle, interpretierte Programmiersprachen Abhilfe.

Eine davon ist die in den vergangenen Jahren sehr populär gewordene Programmiersprache bzw. Programmierumgebung *Scratch*. Anders als bei textuellen Programmiersprachen müssen hier die Programmierer keinen textuellen Sourcecode schreiben. Vielmehr setzen sich Scratch-Programme aus graphischen Bausteinen zusammen (siehe Abbildung 1.1). Die Programmierumgebung unterstützt dabei das Erstellen von Programmen durch einfache Drag-and-Drop-Aktionen mit der Maus. In

Scratch können dem Programmierer keine syntaktischen Fehler mehr unterlaufen und auf einen Compiler kann verzichtet werden. Mehr Informationen zu Scratch findet man im Internet unter http://scratch.mit.edu/.

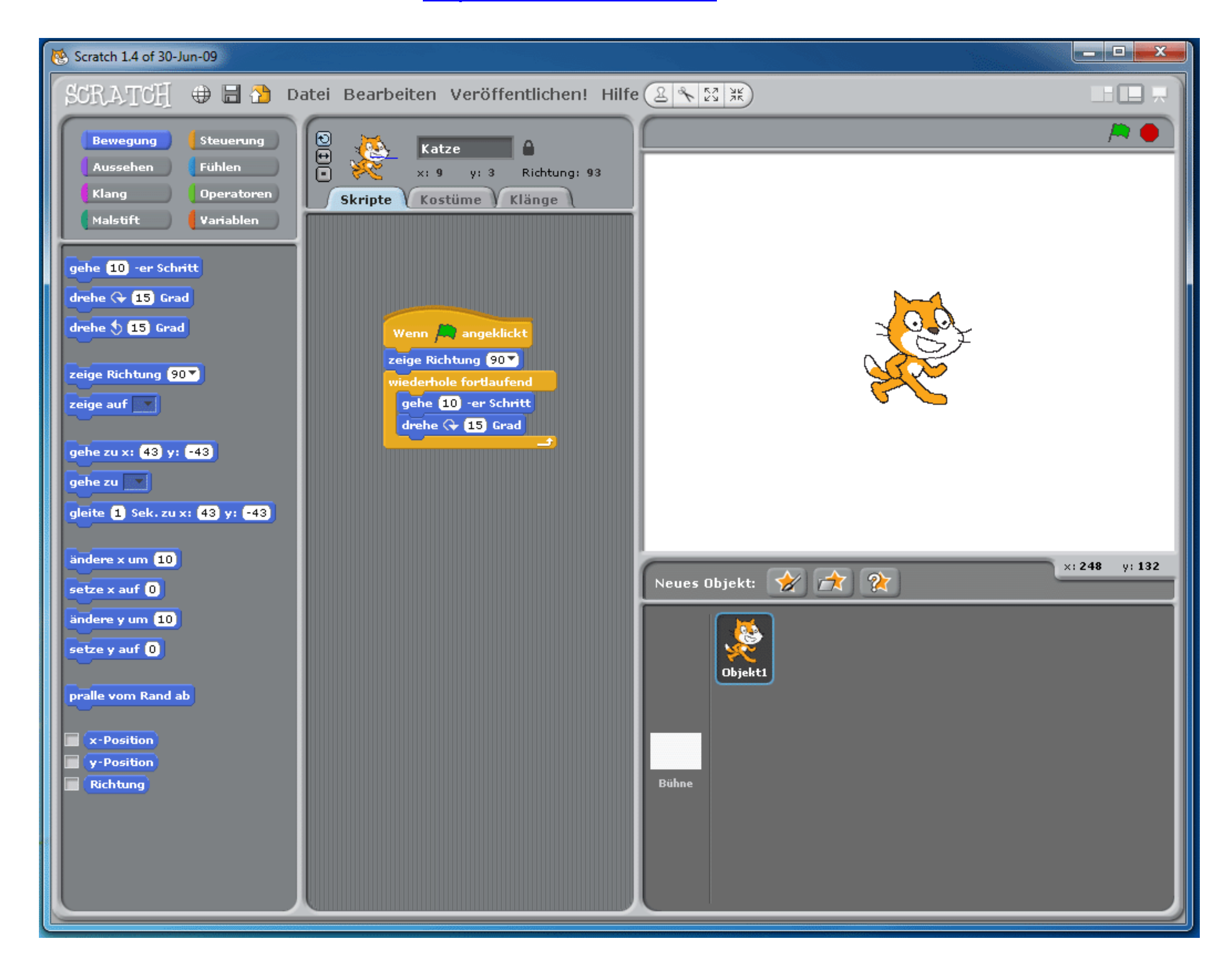


Abb. 1.1: Scratch

Um die Vorteile, die Scratch für Programmieranfänger verspricht, auch denjenigen von Ihnen zugänglich zu machen, die mit Kara, dem Marienkäfer programmieren lernen wollen, haben wir eine Variante von Scratch auf das Kara-Modell übertragen. Was es genau mit diesem so genannten Scratch-Kara-Modell auf sich hat, werden Sie in diesem Handbuch erfahren.

1.3 Der Scratch-Kara-Simulator

Beim Kara-Modell steht nicht so sehr das "Learning-by-Listening" bzw. „Learning-by-Reading“ im Vordergrund, sondern vielmehr das „Learning-by-Doing“, also das

praktische Üben. Genau das ist mit so genannten Simulatoren möglich. Von der Kara-Website kann man sich bspw. einen Java-Kara-Simulator herunterladen und damit Kara-Programme in der Programmiersprache Java schreiben und ausführen. In diesem Handbuch wird der Scratch-Kara-Simulator vorgestellt. Dieser stellt eine Reihe von Werkzeugen zum Erstellen und Ausführen von Scratch-Kara-Programmen zur Verfügung: einen Editor zum Eingeben und Verwalten von Scratch-Kara-Programmen, einen Territoriumsgestalter zum Gestalten und Verwalten von Territorien, einen Interpreter zum Ausführen von Scratch-Kara-Programmen und einen Debugger zum Testen von Scratch-Kara-Programmen. Der Scratch-Kara-Simulator ist einfach zu bedienen, wurde aber funktional und bedienungsmäßig bewusst an professionelle Entwicklungsumgebungen für Java (z.B. Eclipse) angelehnt, um einen späteren Umstieg auf diese zu erleichtern.

1.4 Voraussetzungen

Zielgruppe des Kara-Modells sind Schüler oder Studierende ohne Programmiererfahrung, die die Grundlagen der (imperativen) Programmierung erlernen und dabei ein wenig Spaß haben wollen. Kenntnisse im Umgang mit Computern sind wünschenswert. Der Scratch-Kara-Simulator ist dabei kein Lehrbuch sondern ein Werkzeug, das das Erlernen der Programmierung unterstützt. Auf gute Lehrbücher zum Erlernen der Programmierung wird in Kapitel 8 (Literatur zum Programmieren lernen) hingewiesen.

1.5 Änderungen in Version 1.1 gegenüber Version 1.0

Neben der Beseitigung einiger kleiner Fehler sind insbesondere folgende Erweiterungen in die aktuelle Version 1.1 des Scratch-Kara-Simulators hinzugekommen:

- Der Scratch-Kara-Simulator läuft nun auch mit einem JRE (Java Runtime Environment), es ist nicht mehr unbedingt ein JDK erforderlich.
- Scratch-Programme lassen sich nun ausdrucken.
- Scratch-Programme lassen sich als Bilder abspeichern.
- Die Beschriftungen der Standard-Blöcke lassen sich über Properties ändern.

1.6 Anmerkungen

Der Scratch-Kara-Simulator wurde mit dem Tool „Solist“ (Version 2.1) erstellt. Solist ist eine spezielle Entwicklungsumgebung für Miniprogrammierwelt-Simulatoren. Solist ist ein Werkzeug der Werkzeugfamilie des „Programmier-Theaters“, eine Menge von Werkzeugen zum Erlernen der Programmierung. Metapher aller dieser

Werkzeuge ist die Theaterwelt. Schauspieler (im Falle von Solist „Solisten") agieren auf einer Bühne, auf der zusätzlich Requisiten platziert werden können. Eine Aufführung entspricht der Ausführung eines Programms.

Im Falle des Scratch-Kara-Simulators ist die Bühne das Territorium, auf dem der Marienkäfer als Solist (es gibt nur einen Marienkäfer) agiert. Requisiten sind Pilze, Baumstümpfe und Kleeblätter. Wenn Ihnen also beim Umgang mit dem Scratch-Kara-Simulator der Begriff „Solist" begegnet, kennen Sie nun hiermit den Hintergrund dieser Namenswahl.

Mehr Informationen zu Solist finden Sie im WWW unter www.programmierkurs-java.de/solist.

1.7 Aufbau des Benutzerhandbuch

Das Benutzungshandbuch ist in 8 Kapitel gegliedert. Nach dieser Einleitung wird in Kapitel 2 die Installation des Scratch-Kara-Simulators beschrieben Kapitel 3 stellt ausführlich das Kara-Modell im Allgemeinen und das Scratch-Kara-Modell im Speziellen vor. Eine Übersicht über die Programmierkonzepte der imperativen Programmierung mit dem Scratch-Kara-Modell gibt Kapitel 4. Wie Sie Ihr erstes Scratch-Kara-Programm zum Laufen bringen, erfahren Sie kurz und knapp in Kapitel 5. Dieses enthält eine knappe Einführung in die Elemente und die Bedienung des Scratch-Kara-Simulators. Sehr viel ausführlicher geht dann Kapitel 6 auf die einzelnen Elemente und die Bedienung des Scratch-Kara-Simulators ein. Kapitel 7 demonstriert an einigen Beispielprogrammen die Programmierung mit dem Scratch-Kara-Simulator und gibt Hinweise zu Übungsaufgaben. Kapitel 8 enthält letztendlich Hinweise zu geeigneter Begleitliteratur zum Erlernen der Programmierung im Allgemeinen.

2 Installation

2.1 Voraussetzungen

Voraussetzung zum Starten des Scratch-Kara-Simulators ist ein Ein Java Runtime Environment SE (JRE) oder ein Java Development Kit SE (JDK) der Version 6 oder höher. Das jeweils aktuelle JRE bzw. JDK kann über die Website http://www.oracle.com/technetwork/java/javase/downloads/index.html bezogen werden und muss anschließend installiert werden.

2.2 Download, Installation und Start

Der Kara-Solist-Simulator kann von der Website www.programmierkurs-java.de/solist kostenlos herunter geladen werden. Er befindet sich in einer Datei namens *karasimulator-scratch-1.1.zip*. Diese muss zunächst entpackt werden. Es entsteht ein Ordner namens *karasimulator-scratch-1.1* (der so genannte *Simulator-Ordner*), in dem sich eine Datei *simulator.jar* befindet. Durch Doppelklick auf diese Datei wird der Scratch-Kara-Simulator gestartet. Alternativ lässt er sich auch durch Eingabe des Befehls `java –jar simulator.jar` in einer Console starten.

Beim ersten Start sucht der Scratch-Kara-Simulator nach der JDK-Installation auf Ihrem Computer. Sollte diese nicht gefunden werden, werden Sie aufgefordert, den entsprechenden Pfad einzugeben. Der Pfad wird anschließend in einer Datei namens *solist.properties* im Simulator-Ordner gespeichert, wo er später wieder geändert werden kann, wenn Sie bspw. eine aktuellere JDK-Version auf Ihrem Rechner installieren sollten. In der Property-Datei können Sie weiterhin die Sprache angeben, mit der der Scratch-Kara-Simulator arbeitet. Aktuell wird allerdings nur deutsch unterstützt.

Weiterhin ist es über entsprechende Properties in der Property-Datei möglich, die Beschriftungen der Standard-Blöcke zu ändern.

3 Das (Scratch-)Kara-Modell

Computer können heutzutage zum Lösen vielfältiger Aufgaben genutzt werden. Die Arbeitsanleitungen zum Bearbeiten der Aufgaben werden ihnen in Form von Programmen mitgeteilt. Diese Programme, die von Programmierern entwickelt werden, bestehen aus einer Menge von Befehlen bzw. Anweisungen, die der Computer ausführen kann. Die Entwicklung solcher Programme bezeichnet man als Programmierung.

Das Kara-Modell ist ein spezielles didaktisches Modell zum Erlernen der Programmierung. Im Kara-Modell nimmt ein virtueller Marienkäfer die Rolle des Computers ein. Diesem Marienkäfer können ähnlich wie einem Computer Befehle erteilt werden, die dieser ausführt.

Ihnen als Programmierer werden bestimmte Aufgaben gestellt, die sie durch die Steuerung des Marienkäfers zu lösen haben. Derartige Aufgaben werden im Folgenden Marienkäfer- oder Kara-Aufgaben genannt. Zu diesen Aufgaben müssen Sie so genannte Kara-Programme entwickeln, die die Aufgaben korrekt und vollständig lösen. Die Aufgaben werden dabei nach und nach komplexer. Zum Lösen der Aufgaben müssen bestimmte Programmierkonzepte eingesetzt werden, die im Kara-Modell inkrementell eingeführt werden.

Die Grundidee des Kara-Modells ist ausgesprochen einfach: Sie als Programmierer müssen einen (virtuellen) Marienkäfer durch eine (virtuelle) Landschaft steuern und ihn gegebene Aufgaben lösen lassen.

3.1 Kara-Modell

Kara, der Marienkäfer, lebt in einer rechteckigen Welt, die aus einzelnen Feldern besteht. Er kann sich nur von Feld zu Feld bewegen. Verlässt der Käfer die Welt an einer Seite, so betritt er sie auf der gegenüberliegenden Seite wieder. Auf den Feldern gibt es verschiedene Arten von Objekten (siehe auch Abbildung 3.1):

- Unbewegliche Baumstümpfe: Kara kann Felder mit einem Baumstumpf nicht betreten. Auf einem Feld mit einem Baumstumpf kann kein anderer Gegenstand liegen.
- Kleeblätter: Kara kann beliebig viele Kleeblätter aufnehmen, und er hat einen unerschöpflichen Vorrat an Blättern zum Ablegen. Kara kann auf einem Kleeblatt stehen, aber auf einem Feld kann höchstens ein Kleeblatt liegen.
- Verschiebbare Pilze: Kara kann nicht auf einem Feld stehen, das von einem Pilz belegt ist. Ein Pilz kann auf einem Kleeblatt stehen. Läuft Kara in einen

Pilz, so verschiebt er ihn geradeaus ins nächste Feld. Er ist aber zu schwach, um gleichzeitig zwei Pilze zu verschieben.

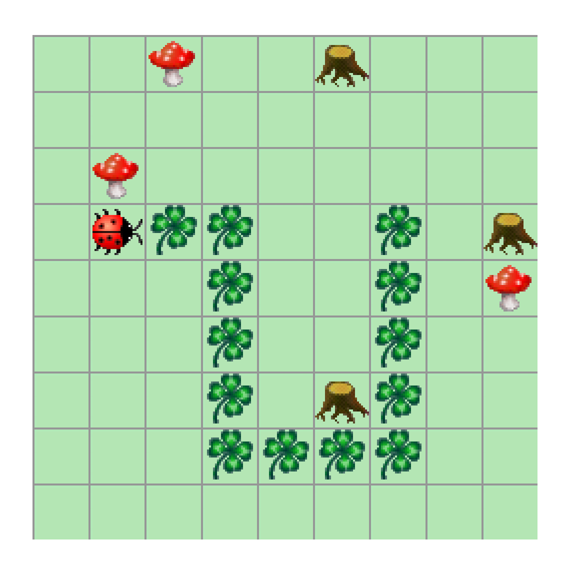


Abb. 3.1: Kara-Landschaft

3.2 Kara-Befehle

Kara kennt fünf Befehle und fünf Testbefehle, durch deren Aufruf ein Programmierer den Marienkäfer durch ein gegebenes Territorium steuern kann. Sie können sich den Marienkäfer quasi als einen virtuellen Prozessor vorstellen, der im Gegensatz zu realen Computer-Prozessoren (zunächst) keine arithmetischen und logischen Operationen ausführen kann, sondern in der Lage ist, mit einem kleinen Grundvorrat an Befehlen ein Territorium zu „erkunden". Diese Befehle werden im Folgenden aufgelistet und im kommenden Kapitel genauer vorgestellt.

- Befehle: Die Ausführung der fünf Befehle, die der Marienkäfer kennt, bewirkt eine Zustandsänderung im Territorium.
    - `move`: Schritt vorwärts
    - `turnLeft`: 90°-Linksdrehung
    - `turnRight`: 90°-Rechtsdrehung
    - `putLeaf`: Kleeblatt hinlegen
    - `removeLeaf`: Kleeblatt aufnehmen

- Testbefehle: Mit Hilfe der fünf Testbefehle ist es möglich, bestimmte Wahrheitswerte, die sich auf den aktuellen Zustand des Territoriums beziehen, abzufragen. Dabei gibt es genau zwei Wahrheitswerte (auch als *boolesche Werte* bezeichnet), nämlich *wahr* (bzw. *true*) und *falsch* (bzw. *false*).
    - `bool treeFront`: Kara vor Baum?
    - `bool treeLeft` : Baum links von Kara?
    - `bool treeRight`: Baum rechts von Kara?
    - `bool onLeaf`: Kara auf Kleeblatt?
    - `bool mushroomFront`: Kara vor Pilz?

4 Imperative Programmierkonzepte des Scratch-Kara-Modells

In diesem Kapitel werden die Befehle, die der Marienkäfer kennt, genauer erläutert. Außerdem enthält dieses Kapitel eine Übersicht über die grundlegende Benutzung des Scratch-Kara-Simulators und die Programmierkonzepte der imperativen Programmierung, die mit der hier vorliegenden Scratch-Variante umsetzbar sind.

Wenn Sie noch überhaupt keine Kenntnisse der Programmierung haben, empfehle ich Ihnen, sich Informationen zu den allgemeinen Grundlagen der Programmierung auf der folgenden Website durchzulesen (Leseprobe des Java-Hamster-Buches): http://www-is.informatik.uni-oldenburg.de/~dibo/hamster/leseprobe/node3.html.

4.1 Aufbau des Editors

Nach dem Starten des Scratch-Kara-Simulators erscheint das in Abbildung 4.1 dargestellte Fenster auf dem Bildschirm. Es besteht aus den folgenden Komponenten:

- Menüleiste
- Toolbar
- Editor-Bereich
- Simulationsbereich
- Meldungsbereich

In diesem Kapitel werden wir uns genauer mit dem Editor-Bereich beschäftigen. Die anderen Komponenten, ihr Zusammenspiel und ihre Benutzung werden ausführlich in Kapitel 6 beschrieben.

Der Editor-Bereich ist wiederum in drei Unterkomponenten unterteilt (siehe Abb. 4.2):

- Kategorienauswahl
- Blockpalette
- Programmbereich

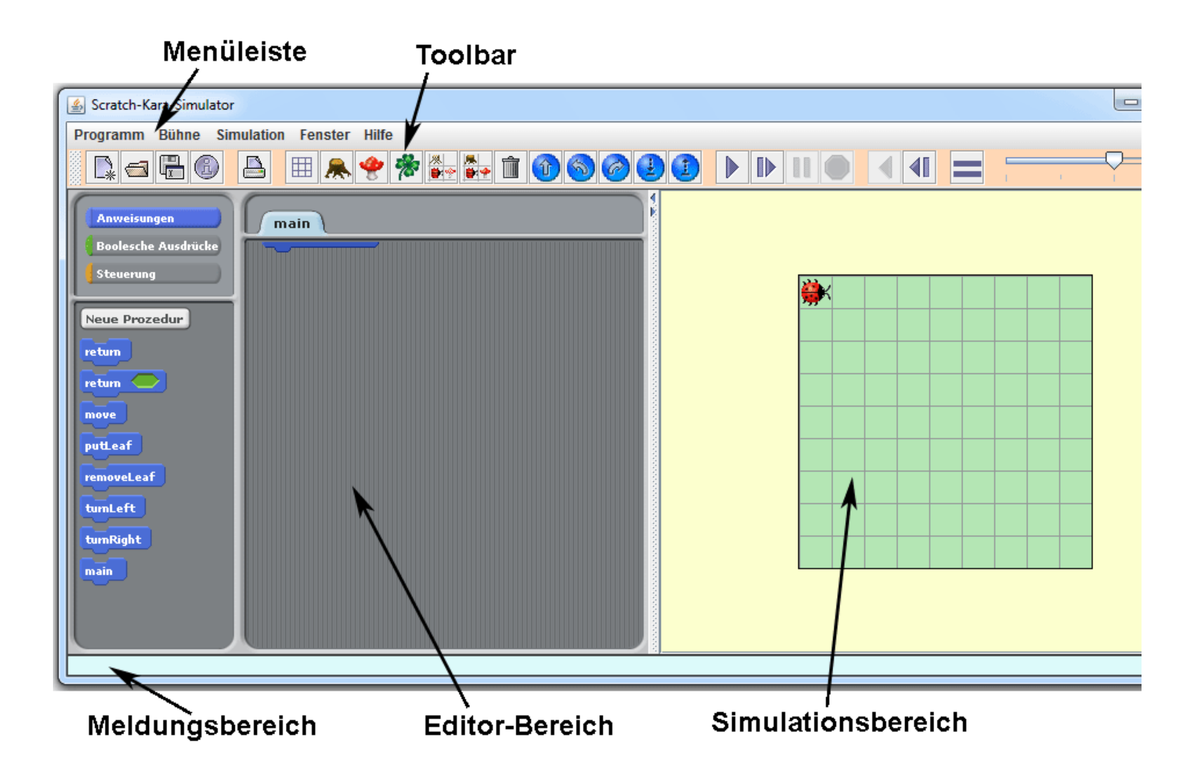


Abb. 4.1: Komponenten des Scratch-Kara-Simulators

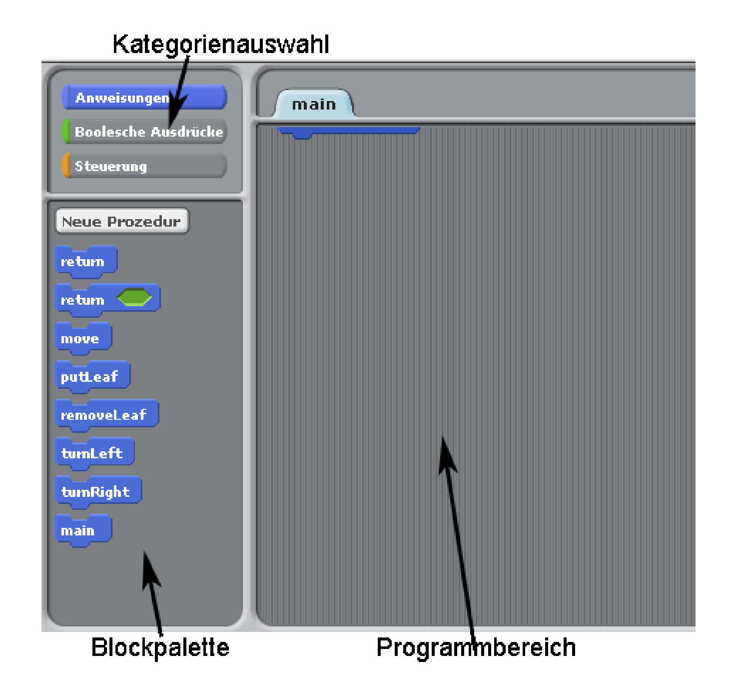


Abb. 4.2: Komponenten des Editors

4.1.1 Kategorienauswahl

In der Kategorienauswahl finden sich die drei Buttons „Anweisungen“, „Boolesche Ausdrücke“ und „Steuerung“. Durch Mausklick auf einen der Buttons erscheinen entsprechende Blöcke der jeweiligen Kategorie in der Blockpalette.

4.1.2 Blockpalette

In der Blockpalette werden die Blöcke der in der Kategorienauswahl aktuell ausgewählten Kategorie angezeigt. Blöcke repräsentieren dabei bestimmte Programmbausteine. Blöcke – genauer gesagt Kopien hiervon – lassen sich per Drag-und-Drop mit der Maus in den Programmbereich ziehen, um Programme zu erstellen. Wählen Sie hierzu den entsprechenden Block, klicken Sie ihn mit der Maus (linke Taste) an, ziehen Sie die Maus bei gedrückter linker Taste in den Programmbereich und lassen Sie die Maustaste los.

Folgende Blöcke werden standardmäßig in der Blockpalette angezeigt (siehe auch die Abbildungen 4.3, 4.4 und 4.5):

- Anweisungsblöcke (Kategorie „Anweisungen“):



Abbildung 4.3: Blockpalette der Kategorie „Anweisungen“

  - `return`: Repräsentiert die return-Anweisung zum Verlassen einer Prozedur (siehe Abschnitt 4.3.3).

- `return <boolescher Ausdruck>`: Repräsentiert die boolesche-return-Anweisung zum Verlassen einer booleschen Funktion (siehe Abschnitt 4.6.1).
- `move`: Repräsentiert den Kara-Grundbefehl *move*.
- `putLeaf`: Repräsentiert den Kara -Grundbefehl *putLeave*.
- `removeLeaf`: Repräsentiert den Kara -Grundbefehl *removeLeaf*.
- `turnLeft`: Repräsentiert den Kara -Grundbefehl *turnLeft*.
- `turnRight`: Repräsentiert den Kara -Grundbefehl *turnRight*.
- `main`: Repräsentiert die main-Prozedur (siehe Abschnitt 4.2.4).
- Über den Button „`Neue Prozedur`“ lassen sich neue Prozeduren definieren (siehe Abschnitt 4.3).

- Boolesche-Ausdrucks-Blöcke (Kategorie „Boolesche Ausdrücke“):



Abbildung 4.4: Blockpalette der Kategorie „Boolesche Ausdrücke“

- `true`: Repräsentiert den Wahrheitswert *wahr*.
- `false`: Repräsentiert den Wahrheitswert *falsch*.
- `not <boolescher Ausdruck>`: Repräsentiert den booleschen Nicht-Operator (siehe Abschnitt 4.4).
- `<boolescher Ausdruck> and <boolescher Ausdruck>`: Repräsentiert den booleschen Und-Operator (siehe Abschnitt 4.4).

- `<boolescher Ausdruck> or <boolescher Ausdruck>`: Repräsentiert den booleschen Oder-Operator (siehe Abschnitt 4.4).
- `mushroomFront`: Repräsentiert den Marienkäfer-Testbefehl *mushroomFront*.
- `onLeaf`: Repräsentiert den Marienkäfer-Testbefehl *onLeaf*.
- `treeFront`: Repräsentiert den Marienkäfer-Testbefehl *treeFront*.
- `treeLeft`: Repräsentiert den Marienkäfer-Testbefehl *treeLeft*.
- `treeRight`: Repräsentiert den Marienkäfer-Testbefehl *treeRight*.
- Über den Button „`Neue Funktion`“ lassen sich neue boolesche Funktionen definieren (siehe Abschnitt 4.6).

- Steuerungsblöcke (Kategorie „Steuerung“):

  Bei den Steuerungsblöcken handelt sich um spezielle Anweisungsblöcke.

  - `if <boolescher Ausdruck>`: Repräsentiert die bedingte Anweisung (siehe Abschnitt 4.5.1).
  - `if <boolescher Ausdruck>-else`: Repräsentiert die Alternativ-Anweisung (siehe Abschnitt 4.5.2).
  - `while <boolescher Ausdruck>`: Repräsentiert die Solange-Schleife (siehe Abschnitt 4.5.3).
  - `do-while <boolescher Ausdruck>`: Repräsentiert die Wiederhole-Solange-Schleife (siehe Abschnitt 4.5.4).

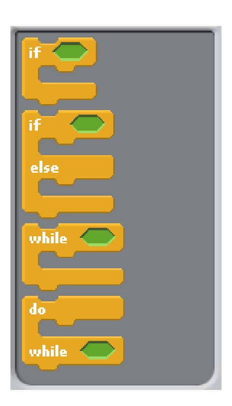


Abbildung 4.5: Blockpalette der Kategorie „Steuerung“

4.1.3 Programmbereich

Im Programmbereich liegen die Scratch-Kara-Programme. Blöcke, die aus der Blockpalette in den Programmbereich gezogen wurden, lassen sich hier per Drag-and-Drop-Aktion weiter hin und herschieben. Blöcke lassen sich aus dem Programmbereich wieder entfernen, indem man sie in die Kategorienauswahl oder die Blockpalette zieht. Alternativ kann man auch oberhalb eines zu entfernenden Blocks ein Popup-Menü aktivieren und hierin das Menü-Item „entfernen“ anklicken.

4.2 Anweisungen und Programme

Programme setzen sich aus einer Menge von Befehlen bzw. Anweisungen zusammen.

4.2.1 Marienkäfer-Befehle

Die Aufgabe eines Kara-Programmierers besteht darin, den Marienkäfer durch eine Landschaft zu steuern, um dadurch gegebene Marienkäfer-Aufgaben zu lösen. Zur Steuerung des Marienkäfers müssen ihm Anweisungen in Form von Befehlen gegeben werden. Der Marienkäfer besitzt dabei die Fähigkeit, vier verschiedene Befehle zu verstehen und auszuführen:

- `move`: Der Marienkäfer hüpft eine Kachel in seiner aktuellen Blickrichtung nach vorne.
- `turnLeft`: Der Marienkäfer dreht sich auf der Kachel, auf der er gerade steht, um 90 Grad entgegen dem Uhrzeigersinn.
- `turnRight`: Der Marienkäfer dreht sich auf der Kachel, auf der er gerade steht, um 90 Grad im Uhrzeigersinn.
- `removeLeaf`: Der Marienkäfer frisst von der Kachel, auf der er sich gerade befindet, ein Kleeblatt.
- `putLeaf`: Der Marienkäfer legt auf der Kachel, auf der er sich gerade befindet, ein Kleeblatt aus seinem Maul ab.

Wie Sie vielleicht schon festgestellt haben, können bei einigen Befehlen Probleme auftreten, bspw. wenn Sie dem Marienkäfer den Befehl move geben, sich aber vor ihm ein Baumstumpf befindet. Bringen Sie den Marienkäfer in diese für ihn unlösbaren Situationen, dann ist der Marienkäfer derart von Ihnen enttäuscht, dass er im Folgenden nicht mehr bereit ist, weitere Befehle auszuführen. Derartige Fehler werden Laufzeitfehler genannt. Laufzeitfehler treten während der Ausführung eines Programmes auf. Programme, die zu Laufzeitfehlern führen können, sind nicht korrekt!

4.2.2 Anweisungen

In imperativen Programmiersprachen werden Verarbeitungsvorschriften durch so genannte Anweisungen ausgedrückt. Anweisungen, die nicht weiter zerlegt werden können, werden elementare Anweisungen genannt. In der Marienkäfer-Sprache sind die fünf Grundbefehle elementare Anweisungen.

4.2.3 Anweisungssequenzen

Eine Folge von Anweisungen, die nacheinander ausgeführt werden, wird als Anweisungssequenz bezeichnet. Die einzelnen Anweisungen einer Anweisungssequenz werden von oben nach unten hintereinander ausgeführt.

Befindet sich im Programmbereich des Editors ein Anweisungsblock A und zieht man einen anderen Anweisungsblock B in die Nähe (ober- und unterhalb), dann erscheint irgendwann ein transparenter grauer Bereich (siehe Abbildung 4.6). Lässt man B nun los, schnappen Zapfen (unten an den Anweisungsblöcken) und Mulde (oben an den Anweisungsblöcken) der beiden Blöcke ein (man spricht auch von „andocken“) und A und B bilden nun eine Anweisungssequenz (in Scratch wird der Begriff „Stapel“ verwendet). Zieht man im Folgenden den obersten Block einer Anweisungssequenz im Programmbereich mit der Maus hin und her, wird automatisch der gesamte Block mit verschoben. Eine Anweisungssequenz kann man wieder trennen, indem man einen mittleren Block verschiebt. Darunter liegende Blöcke werden dann mit verschoben, darüber liegende Blöcke bleiben liegen und werden abgetrennt.

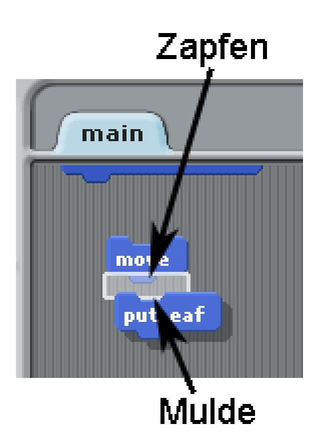


Abbildung 4.6: Bilden von Anweisungssequenzen

4.2.4 Programme

Scratch-Kara-Programme starten immer durch Ausführung der so genannten *main-Prozedur*. Ausgeführt wird dabei die Anweisungssequenz, die mit der main-Prozedur verbunden ist. Hierzu besitzt die Prozedur am oberen Rand des Programmbereichs einen Zapfen (den so genannten *Prozedurzapfen*). Hier muss man die entsprechende Anweisungssequenz andocken (siehe Abbildung 4.7).

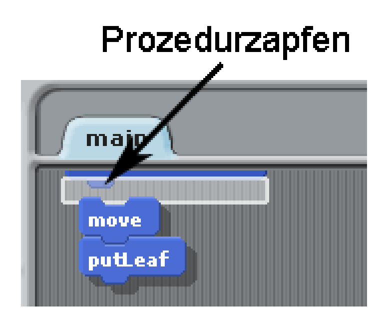


Abbildung 4.7: Scratch-Kara-Programme

4.2.5 Beispielprogramm

Das in Abbildung 4.8 dargestellte Kara-Programm bewirkt, dass der Marienkäfer in dem in Abbildung 4.9 skizzierten Territorium zwei Kleeblätter frisst:

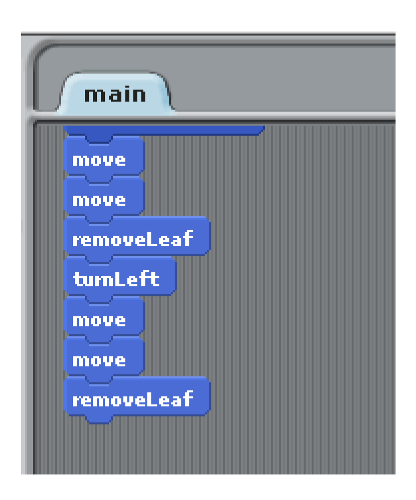


Abb. 4.8: Beispielprogramm 1

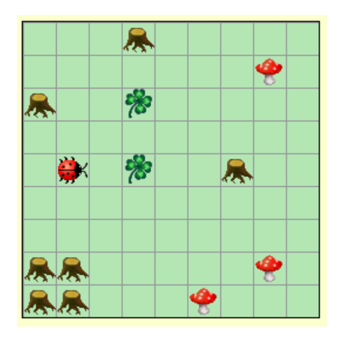


Abb. 4.9: Territorium zu Beispielprogramm 1

4.3 Prozeduren

Prozeduren dienen zur Vereinbarung neuer Befehle. Diesbezüglich sind zwei Aspekte zu betrachten: die Definition von Prozeduren und deren Aufruf, d.h. Ausführung.

4.3.1 Prozedurdefinition

Durch eine Prozedurdefinition wird ein neuer Befehl vereinbart. In der Definition muss zum einen angegeben werden, wie der Befehl heißt (Prozedurname), und zum anderen muss festgelegt werden, was der Marienkäfer tun soll, wenn er den neuen Befehl erhält (Prozedurinhalt).

Eine neue Prozedur können Sie definieren, indem Sie in der Blockpalette (Kategorie „Anweisungen“) auf den Button „Neue Prozedur“ klicken. Über ein Dialogfenster werden Sie anschließend nach dem Namen der neuen Prozedur gefragt. Nach dem Anklicken des OK-Buttons der Dialogbox wird die neue Prozedur angelegt und im Programmbereich erscheint ein neuer Tab-Button mit dem Namen der neuen Prozedur. Den Prozedurinhalt, d.h. die Anweisungen, die ausgeführt werden sollen, wenn die Prozedur aufgerufen wird, können sie analog zu der main-Prozedur definieren. In Abbildung 4.10 sehen Sie die Definition einer Prozedur namens „turnBack“.



Abb. 4.10: Prozedurdefinition

Die main-Prozedur ist übrigens eine besondere Prozedur. Sie wird automatisch beim Start eines Programms aufgerufen.

Durch Anklicken eines Tab-Buttons im Programmbereich wird im Programmbereich der Programminhalt der entsprechenden Prozedur angezeigt. Oberhalb des Tab-Buttons können Sie weiterhin ein Popup-Menü aktivieren, über das sich die jeweilige Prozedur schließen, umbenennen und löschen lässt.

Alternativ lässt sich eine Prozedur auch dadurch definieren, dass oberhalbs eines Blockes im Programmbereich ein Popup-Menü aktiviert wird. Wird hierin das Item „In Prozedur auslagern“ ausgewählt, wird eine neue Prozedur definiert, in der automatisch der Block und alle ihm unter Umständen folgenden Blöcke der Anweisungssequenz ausgelagert werden. Die ausgelagerten Blöcke werden von ihrer ursprünglichen Position entfernt und dort durch einen Block ersetzt, der die neue Prozedur repräsentiert.

4.3.2 Prozeduraufruf

Durch eine Prozedurdefinition wird ein neuer Befehl eingeführt. Ein Aufruf des neuen Befehls wird Prozeduraufruf genannt. Nachdem Sie eine neue Prozedur angelegt haben, erscheint in der Blockpalette (Kategorie „Anweisungen) automatisch ein neuer Block mit dem von Ihnen gewählten Prozedurnamen. Diesen Block können Sie zur Gestaltung Ihrer Programme nun genauso nutzen wie die anderen Blöcke.

Durch Doppelklick auf einen Prozedurblock wird die entsprechende Prozedur im Programmbereich geöffnet und angezeigt. Oberhalbs des Prozedurblockes können Sie weiterhin ein Popup-Menü aktivieren, über das sich die jeweilige Prozedur öffnen, umbenennen und löschen lässt.

Wird bei der Ausführung eines Programms ein Block erreicht, der eine Prozedur repräsentiert, wird diese Prozedur aufgerufen. Das bedeutet, es wird die Anweisungssequenz ausgeführt, die mit dem Prozedurzapfen der Prozedur verbunden ist. Der Kontrollfluss des Programms verzweigt beim Prozeduraufruf in den Rumpf der Prozedur, führt die dortigen Anweisungen aus und kehrt nach der Abarbeitung der letzten Anweisung des Rumpfes an die Stelle des Prozeduraufrufs zurück.

4.3.3 Return-Anweisung

In der Blockpalette (Kategorie „Anweisungen“) gibt es noch zwei besondere Blöcke: `return` und `return <boolescher Ausdruck>`. Diese können genauso wie die anderen Blöcke auch zur Programmgestaltung eingesetzt werden. Wird bei der

Ausführung einer Prozedur ein entsprechender Block erreicht, wird die Prozedur unmittelbar verlassen.

Da Anweisungen, die in einer Anweisungssequenz unmittelbar nach einer return-Anweisung stehen würden, eh nie erreicht werden können, ist es übrigens nicht möglich, andere Blöcke unterhalb eines return-Blockes anzudocken.

4.3.4 Beispielprogramm

Im Scratch-Kara-Programm in Abbildung 4.11 wird die in Abbildung 4.10 definierte Prozedur `turnBack` aufgerufen. Sie bewirkt, dass sich der Marienkäfer zweimal nach links dreht, was einer Drehung um 180 Grad entspricht. Der Marienkäfer frisst daher in dem in Abbildung 4.12 skizzierten Territorium die beiden Körner:

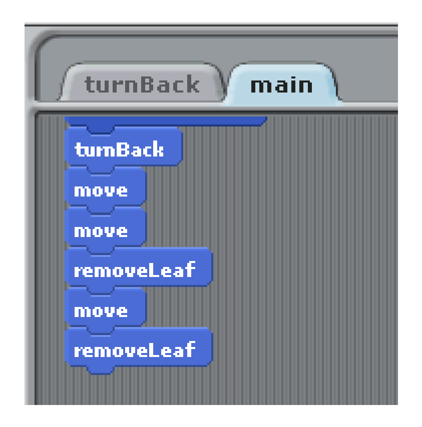


Abb. 4.11: Beispielprogramm 2

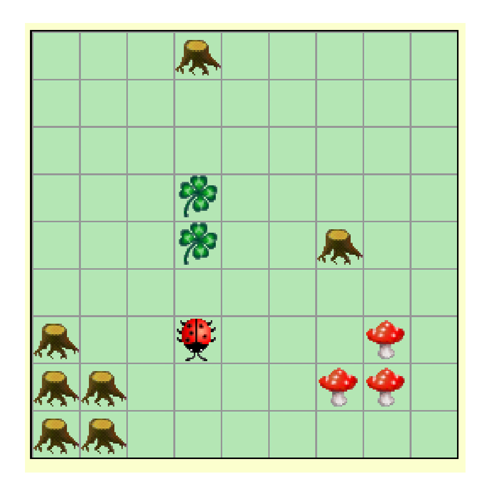


Abb. 4.12: Marienkäfer-Territorium zum Beispielprogramm

4.4 Boolesche Ausdrücke

Um Laufzeitfehler zu vermeiden, die ja bspw. dadurch entstehen können, dass Sie dem Marienkäfer den Befehl `move` geben, obwohl er vor einem Baumstumpf steht, existieren fünf so genannte Testbefehle bzw. boolesche Ausdrücke. Testbefehle liefern Wahrheitswerte (auch als boolesche Werte bezeichnet), also *wahr* (`true`) oder *falsch* (`false`):

- `mushroomFront`: Repräsentiert den Marienkäfer-Testbefehl *mushroomFront*.
- `onLeaf`: Liefert den Wert `true`, falls auf der Kachel, auf der der Marienkäfer gerade steht, ein Kleeblatt liegt. Befindet sich kein Kleeblatt auf der Kachel, dann wird der Wert `false` geliefert.
- `treeFront`: Liefert den Wert `true`, falls sich auf der Kachel in Blickrichtung vor dem Marienkäfer ein Baumstumpf befindet. Ist die Kachel nicht durch einen Baumstumpf blockiert, dann wird der Wert `false` geliefert.
- `treeLeft`: Liefert den Wert `true`, falls sich auf der Kachel links vom Marienkäfer ein Baumstumpf befindet. Ist die Kachel nicht durch einen Baumstumpf blockiert, dann wird der Wert `false` geliefert.
- `treeRight`: Liefert den Wert `true`, falls sich auf der Kachel rechts vom Marienkäfer ein Baumstumpf befindet. Ist die Kachel nicht durch einen Baumstumpf blockiert, dann wird der Wert `false` geliefert.

Die fünf Testbefehle und die beiden Wahrheitswerte *true* und *false* werden in der Blockpalette des Editors durch entsprechende Boolesche-Ausdrucks-Blöcke in der Kategorie „Boolesche Ausdrücke“ repräsentiert.

Die fünf Testbefehle und die beiden Wahrheitswerte stellen einfache boolesche Ausdrücke dar. Ausdrücke sind Verarbeitungsvorschriften, die einen Wert berechnen und liefern. Boolesche Ausdrücke liefern einen booleschen Wert. Durch die Verknüpfung boolescher Ausdrücke mittels boolescher Operatoren lassen sich zusammengesetzte boolesche Ausdrücke bilden. Das Scratch-Kara-Modell stellt drei boolesche Operatoren zur Verfügung:

- Der unäre Operator `not` (Negation) negiert den Wahrheitswert seines Operanden, d.h. er dreht ihn um. Liefert ein boolescher Ausdruck `bA` den Wert `true`, dann liefert der boolesche Ausdruck `not bA` den Wert `false`. Umgekehrt gilt, liefert `bA` den Wert false, dann liefert `not bA` den Wert `true`.

- Der binäre Operator `and` (Konjunktion) konjugiert die Wahrheitswerte seiner beiden Operanden, d.h. er liefert genau dann den Wahrheitswert `true`, wenn beide Operanden den Wert `true` liefern.
- Der binäre Operator `or` (Disjunktion) disjungiert die Wahrheitswerte seiner beiden Operanden, d.h. er liefert genau dann den Wahrheitswert `true`, wenn einer seiner Operanden oder seine beiden Operanden den Wert `true` liefern.

Blöcke der Kategorie „boolesche Ausdrücke“ unterscheiden sich in Form und Farbe (grün) von Blöcken der Kategorien „Anweisungen“ (blau) und „Steuerung“ (gelb). Sie lassen sich auch nicht an Anweisungsblöcke andocken. Vielmehr werden sie zur Umsetzung konkreter Kontrollstrukturen (siehe Abschnitt 4.5) eingesetzt. Dazu ist es möglich, sie an die Platzhalter für Bedingungen der Kontrollstrukturen (deren grüne Komponenten) anzudocken.

4.5 Kontrollstrukturen

Kontrollstrukuren dienen zur Manipulation des Kontrollflusses innerhalb eines Programms. Bei den Kontrollstrukturen handelt es sich um spezielle, so genannte zusammengesetzte Anweisungen, die aus mehreren Komponenten bestehen. Die im Scratch-Kara-Simulator vorhandenen Kontrollstrukturen werden in der Blockpalette bei Auswahl der Kategorie „Steuerung“ angezeigt. Die entsprechenden Blöcke können wie die Blöcke der Kategorie „Anweisungen“ genutzt und auch beliebig mit diesen gekoppelt werden.

4.5.1 Bedingte Anweisung

Die bedingte Anweisung, die auch if-Anweisung genannt wird, besteht aus zwei Unterkomponenten: einem booleschen Ausdruck (Bedingung) und einer Anweisungssequenz (true-Teil) (siehe Abbildung 4.13). Hier lassen sich Boolesche-Ausdrucks-Blöcke bzw. Anweisungsblöcke andocken.

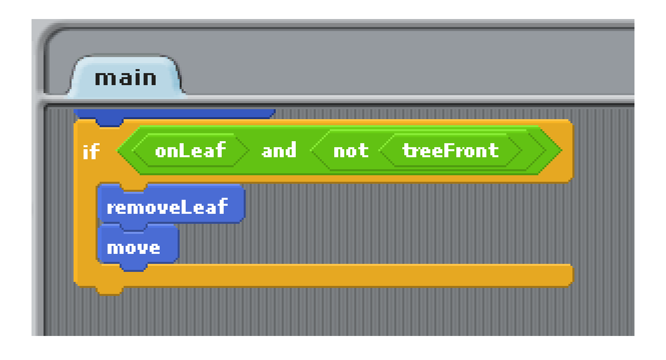


Abb. 4.13: Bedingte Anweisung

Beim Ausführen einer bedingten Anweisung wird zunächst der boolesche Ausdruck ausgewertet. Falls dieser Ausdruck den Wert `true` liefert, d.h. die Bedingung erfüllt ist, werden die Blöcke des true-Teils ausgeführt. Liefert der boolesche Ausdruck den Wert `false`, dann werden die Blöcke des true-Teils nicht ausgeführt.

Im Beispiel in Abbildung 4.13 wird also zunächst überprüft, ob sich auf der Kachel, auf der der Marienkäfer gerade steht, ein Kleeblatt befindet und sich gleichzeitig auf der Kachel vor ihm kein Baumstumpf befindet. Dann (und nur dann) nimmt der Marienkäfer das Kleeblatt und hüpft eine Kachel nach vorne.

4.5.2 Alternativanweisung

Die Alternativanweisung ist eine bedingte Anweisung mit einem angehängten so genannten else-Teil, auch false-Teil genannt (siehe Abbildung 4.14).

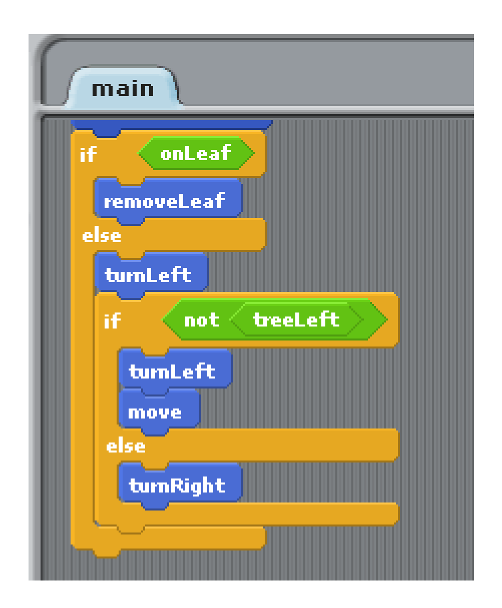


Abb. 4.14: Alternativanweisung

Wird eine Alternativanweisung ausgeführt, dann wird zunächst der Wert der Bedingung (boolescher Ausdruck) ermittelt. Ist die Bedingung erfüllt, d.h. liefert der boolesche Ausdruck den Wert `true`, dann werden die die Blöcke des true-Teils ausgeführt. Liefert der boolesche Ausdruck den Wert `false`, dann werden die Blöcke des false-Teils ausgeführt.

Im Beispiel in Abbildung 4.14 wird zunächst überprüft, ob sich der Marienkäfer auf einer Kachel mit einem Kleeblatt befindet. Falls das der Fall ist, nimmt er das Kleeblatt und das Programm ist beendet. Falls nicht, dreht er sich linksum. Dann

überprüft er, ob sich links von ihm kein Baumstumpf befindet. Falls das der Fall ist, dreht er sich links und hüpft eine Kachel nach vorne, ansonsten dreht er sich wieder nach rechts.

4.5.3 While-Anweisung

Die while-Anweisung ist eine so genannte Wiederholungsanweisung – auch Schleifenanweisung genannt. Wiederholungsanweisungen dienen dazu, bestimmte Anweisungen mehrmals ausführen zu lassen, solange eine bestimmte Bedingung erfüllt wird.

Die while-Anweisung besteht aus zwei Unterkomponenten: einem booleschen Ausdruck (Schleifenbedingung) und einer Anweisungssequenz (Interationsanweisung) (siehe Abbildung 4.15). Hier lassen sich Boolesche-Ausdrucks-Blöcke bzw. Anweisungsblöcke andocken.

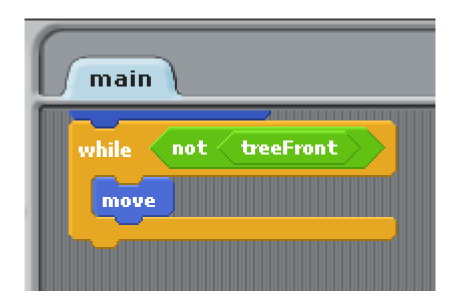


Abb. 4.15: while-Anweisung

Bei der Ausführung einer while-Anweisung wird zunächst überprüft, ob die Schleifenbedingung erfüllt ist, d.h. ob der boolesche Ausdruck den Wert `true` liefert. Falls dies nicht der Fall ist, ist die while -Anweisung unmittelbar beendet. Falls die Bedingung erfüllt ist, wird die Iterationsanweisung einmal ausgeführt. Anschließend wird die Schleifenbedingung erneut ausgewertet. Falls sie immer noch erfüllt ist, wird die Iterationsanweisung ein weiteres Mal ausgeführt. Dieser Prozess (Überprüfung der Schleifenbedingung und falls diese erfüllt ist, Ausführung der Iterationsanweisung) wiederholt sich so lange, bis (hoffentlich) irgendwann einmal die Bedingung nicht mehr erfüllt ist.

Im Beispiel in Abbildung 4.15 läuft der Marienkäfer bis zum nächsten Baumstumpf.

4.5.4 Do-while-Anweisung

Bei Ausführung der while-Anweisung kann es vorkommen, dass die Iterationsanweisung kein einziges Mal ausgeführt wird; nämlich genau dann, wenn

die Schleifenbedingung direkt beim ersten Test nicht erfüllt ist. Für solche Fälle, bei denen die Iterationsanweisung auf jeden Fall mindestens einmal ausgeführt werden soll, existiert die do-while-Anweisung.

Genauso wie die while-Anweisung besteht die do-while-Anweisung aus zwei Unterkomponenten: einem booleschen Ausdruck (Schleifenbedingung) und einer Anweisungssequenz (Interationsanweisung) (siehe Abbildung 4.16). Hier lassen sich Boolesche-Ausdrucks-Blöcke bzw. Anweisungsblöcke andocken.

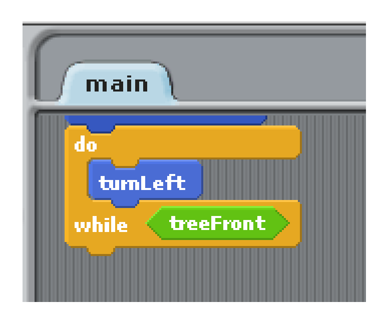


Abb. 4.15: do-while-Anweisung

Bei der Ausführung einer do-while-Anweisung wird zunächst einmal die Iterationsanweisung ausgeführt. Anschließend wird die Schleifenbedingung überprüft. Ist sie nicht erfüllt, d.h. liefert der boolesche Ausdruck den Wert `false`, dann endet die do-while-Anweisung. Ist die Bedingung erfüllt, wird die Iterationsanweisung ein zweites Mal ausgeführt und danach erneut die Schleifenbedingung ausgewertet. Dieser Prozess wiederholt sich so lange, bis irgendwann einmal die Schleifenbedingung nicht mehr erfüllt ist.

Im Beispiel in Abbildung 4.15 dreht sich der Marienkäfer solange linksum, wie sich ein Baumstumpf vor ihm befindet.

4.5.5 Beispielprogramm

Aufgabe: Der Marienkäfer soll bis zum nächsten Baumstumpf oder Pilz laufen und dabei alle Kleeblätter einsammeln.

Lösung: siehe Abbildung 4.16

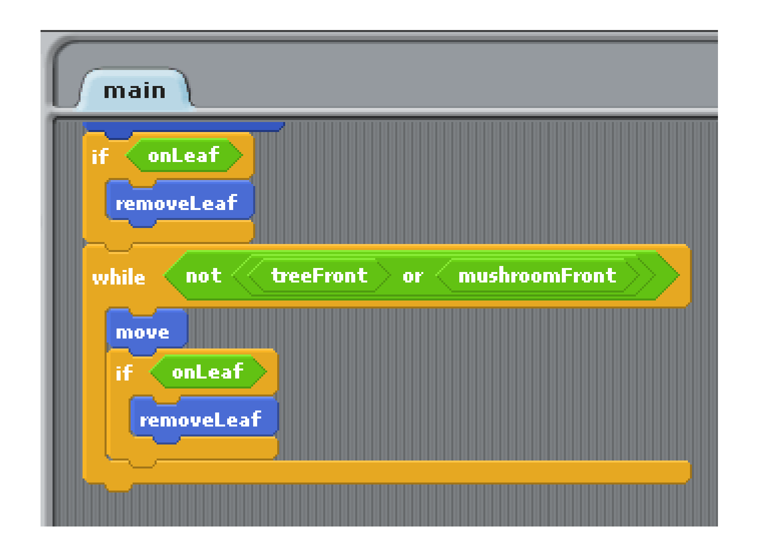


Abb. 4.16: Beispielprogramm zur Kontrollstrukturen

4.6 Boolesche Funktionen

Während Prozeduren dazu dienen, den Befehlsvorrat des Marienkäfers zu erweitern, dienen boolesche Funktionen dazu, neue Testbefehle einzuführen. Da das Scratch-Kara-Modell ausschließlich boolesche Funktionen unterstützt, werden boolesche Funktionen hier einfach als Funktionen bezeichnet.

4.6.1 Boolesche return-Anweisung

Boolesche return-Anweisungen werden in booleschen Funktionen zum Liefern eines booleschen Wertes benötigt. Sie werden durch den Block `return <boolescher Ausdruck>` (Kategorie „Anweisungen") repräsentiert.

Die Ausführung einer booleschen return-Anweisung während der Ausführung einer booleschen Funktion führt zur unmittelbaren Beendigung der Funktionsausführung. Dabei wird der Wert des booleschen Ausdrucks ermittelt und als so genannter Funktionswert zurückgegeben.

4.6.2 Funktionsdefinition

Die Definition einer neuen Funktion erfolgt analog zur Definition einer neuen Prozedur (siehe Abschnitt 4.3). Drücken Sie dazu auf den Button „Neue Funktion" in der Blockpalette zur Kategorie „boolesche Ausdrücke".

Eine neu definierte Funktion wird in der Blockpalette zur Kategorie „boolesche Ausdrücke“ durch einen entsprechenden Boolescher-Ausdrucks-Block repräsentiert und kann überall dort eingesetzt werden, wo auch andere Boolesche-Ausdrucks-Blöcke verwendet werden können (siehe auch Abbildung 4.17).

4.6.3 Funktionsaufruf

Wird bei der Berechnung eines booleschen Ausdrucks eine Funktion aufgerufen, so wird in deren Funktionsrumpf verzweigt und es werden die dortigen Anweisungen aufgerufen. Wird dabei eine boolesche return-Anweisung ausgeführt, so wird der Funktionsrumpf unmittelbar verlassen und an die Stelle des Funktionsaufrufs zurückgesprungen. Der von der booleschen return-Anweisung gelieferte Wert (also der Funktionswert) wird dabei zur Berechnung des booleschen Ausdrucks weiterverwendet.

Endet die Ausführung einer Funktion, ohne dass eine boolesche return-Anweisung erreicht wurde, so wird automatisch der Wert *true* zurückgeliefert. Wird bei der Ausführung einer Funktion eine normale return-Anweisung ausgeführt, dann wird die Funktion unmittelbar verlassen und ebenfalls der Wert *true* geliefert.

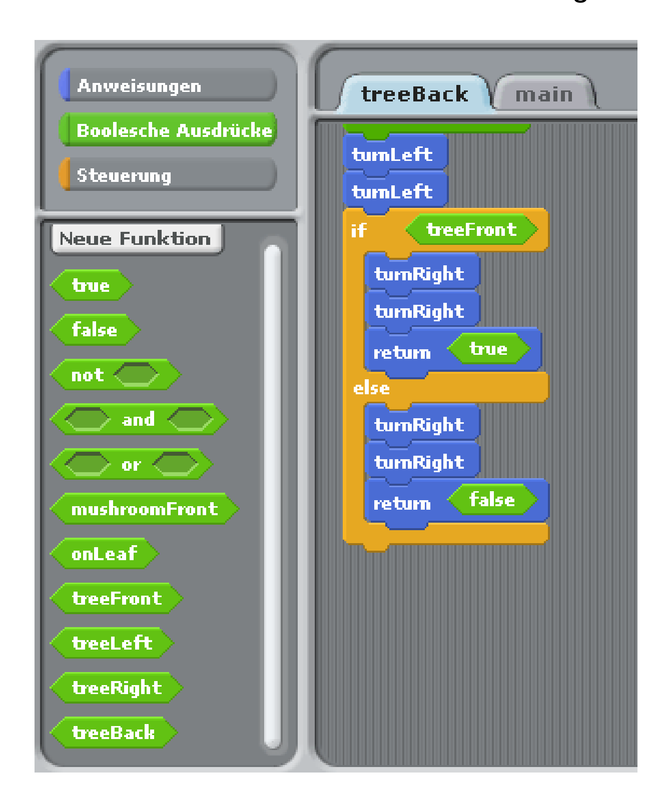


Abb. 4.17: Funktionsdefinition und -aufruf

4.7 Rekursion

Prozeduren bzw. Funktionen, die sich selbst aufrufen, bezeichnet man als rekursive Prozeduren bzw. Funktionen. Der Aufruf der rekursiven Funktion `backAndForth` im folgenden Programm bewirkt, dass der Marienkäfer bis zum nächsten Baumstumpf oder Pilz und anschließend zurück zu seinem Ausgangspunkt läuft.

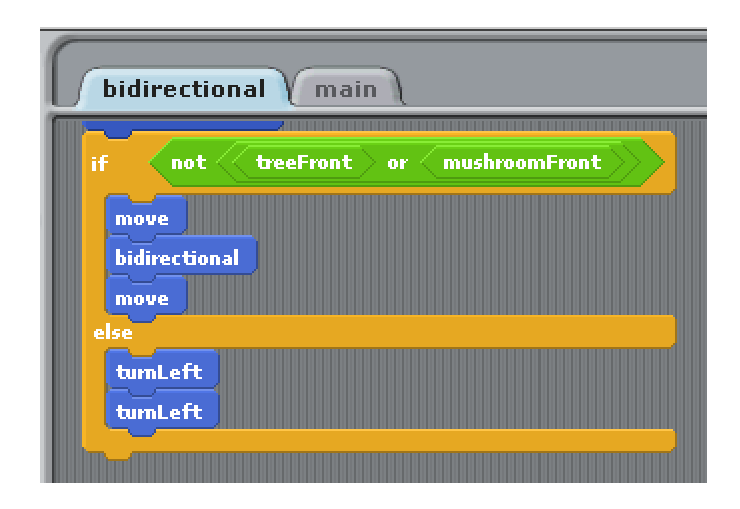


Abb. 4.17: Rekursion

5 Ihr erstes Scratch-Kara-Programm

Nachdem Sie den Scratch-Kara-Simulator gestartet haben, öffnet sich ein Fenster, das in etwa dem in Abbildung 5.1 dargestellten Fenster entspricht. Ganz oben enthält es eine Menüleiste mit 5 Menüs, darunter eine Toolbar mit einer Reihe von Buttons und ganz unten einen Meldungsbereich, in dem Meldungen ausgegeben werden. Im linken Teil befindet sich der Editor-Bereich, im rechten Teil der Simulationsbereich. Im Editor-Bereich geben Sie Scratch-Kara-Programme ein und im Simulationsbereich führen Sie Scratch-Kara-Programme aus.

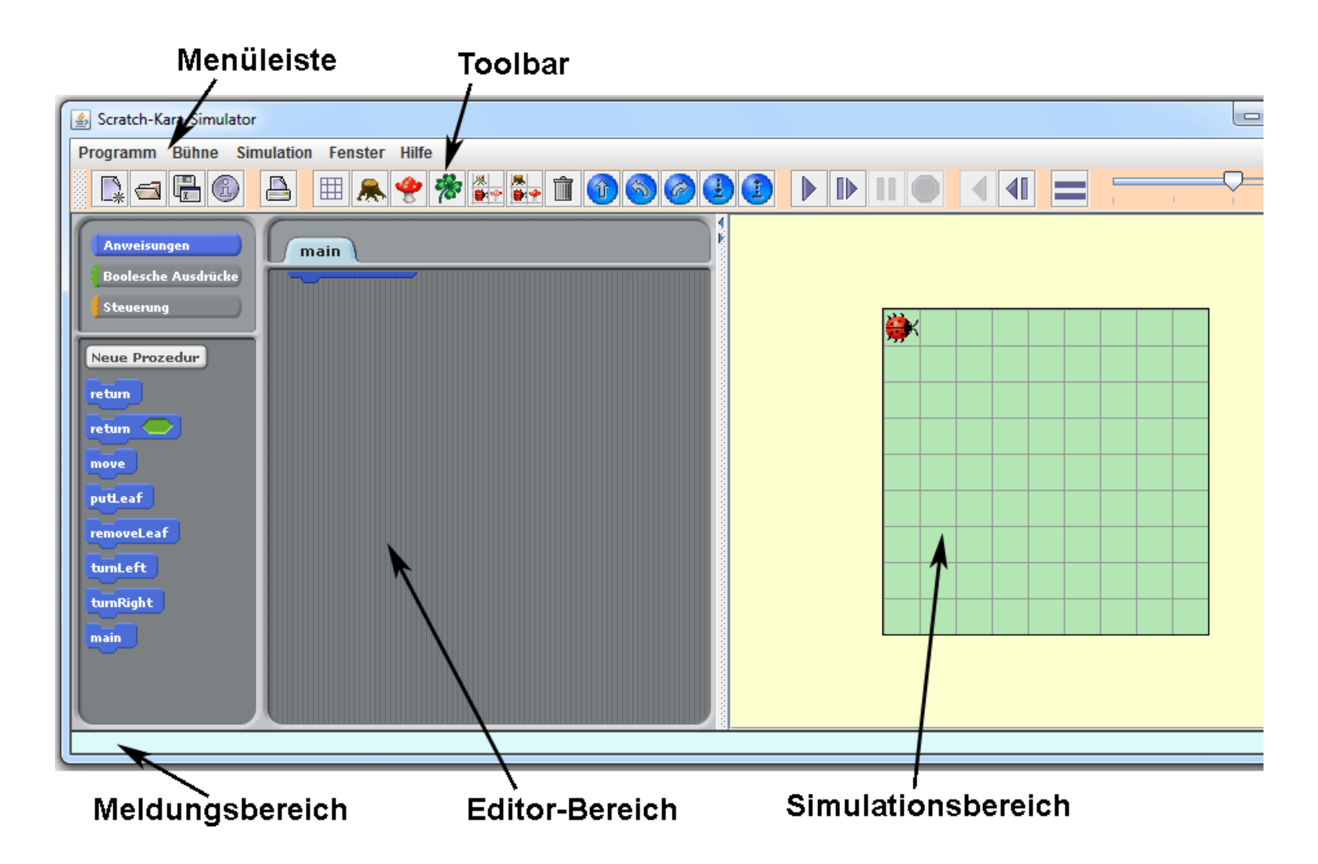


Abb. 5.1: Scratch-Kara-Simulator nach dem Öffnen

Um den Marienkäfer ein wenig näher kennen zu lernen, empfehle ich Ihnen, als erstes zunächst die Kara-Befehle einmal auszuprobieren. Wie das geht, wird in Abschnitt 5.1 erläutert. Anschließend wird in den darauf folgenden Abschnitten im Detail beschrieben, was Sie machen müssen, um Ihr erstes Scratch-Kara-Programm zu schreiben und auszuführen. Insgesamt müssen/können fünf Stationen durchlaufen werden:

- Gestaltung eines Territoriums
- Eingeben eines Scratch-Kara-Programms
- Ausführen eines Scratch-Kara-Programms
- Debuggen eines Scratch-Kara-Programms

5.1 Ausprobieren der Kara-Befehle

Klicken Sie im Menü „Fenster“ das Menü-Item „Befehlsfenster“ an. Es öffnet sich ein Fenster mit dem Titel „Befehlsfenster“. In diesem Fenster erscheinen alle Befehle, die der Marienkäfer kennt.

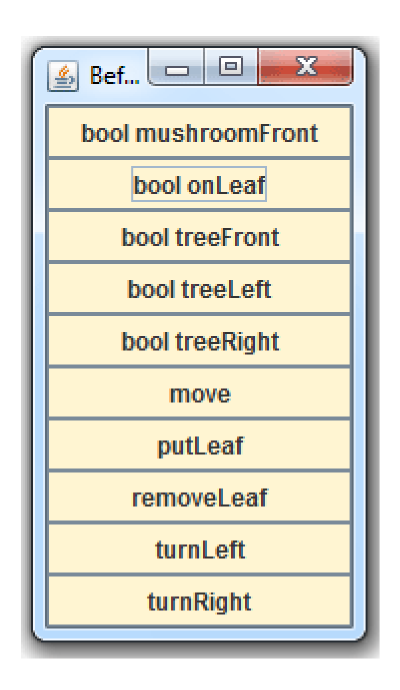


Abb. 5.2: Befehlsfenster

Sie können die jeweiligen Befehle ausführen, indem Sie den Maus-Cursor auf den entsprechenden Button verschieben und diesen anklicken. Klicken Sie bspw. auf den Button „move“, dann hüpft der Marienkäfer in seiner aktuellen Blickrichtung eine Kachel nach vorne (oder es erscheint eine Fehlermeldung, wenn der Marienkäfer bspw. vor einem Baumstumpf steht).

5.2 Gestaltung eines Territoriums

Nun wollen wir ein Territorium aufbauen, in dem unser erstes Programm ablaufen soll. Das geschieht im Simulationsbereich. Hier wird das Territorium dargestellt. Zur Gestaltung des Territoriums müssen Sie die Buttons 6 – 17 (von links) in der so

genannten *Toolbar* benutzen. Diese werden auch als *Gestaltungsbuttons* bezeichnet. Fahren Sie einfach mal mit der Maus über die einzelnen Buttons der Toolbar, dann erscheint jeweils ein Tooltipp, der beschreibt, wozu dieser Button dient.

Zunächst werden wir die Größe des Territoriums anpassen. Klicken Sie dazu auf den Button „Territoriumsgröße ändern“ (6. Button von links). Es erscheint eine Dialogbox, in der Sie die gewünschte Anzahl an Reihen und Spalten eingeben können. Um die dort erscheinenden Werte (jeweils 9) ändern zu können, klicken Sie mit der Maus auf das entsprechende Eingabefeld. Anschließend können Sie den Wert mit der Tastatur eingeben. Nach der Eingabe der Werte klicken Sie bitte auf den OK-Button. Die Dialogbox schliesst sich und das Territorium erscheint in der angegebenen Größe.

Nun werden wir den Marienkäfer, der immer im Territorium sitzt, umplatzieren. Dazu klicken wir den Marienkäfer an und ziehen (man spricht auch von „draggen“) ihn mit gedrückter Maustaste, auf die gewünschte Kachel. Dann lassen wir die Maustaste los.

Nun wollen wir auf einigen Kacheln Baumstümpfe platzieren. Hierzu dient der Button „Baumstumpf setzen“ (7. Button von links). Wenn Sie ihn mit der Maus anklicken, erscheint am Maus-Cursor ein Baumstumpf-Symbol. Bewegen Sie die Maus nun über die Kachel, auf der ein Baumstumpf platziert werden soll, und klicken Sie dort die Maus. Auf der Kachel erscheint nun ein Baumstumpf-Symbol. Baumstümpfe können nicht auf Kacheln platziert werden, auf denen sich bereits andere Elemente befinden.

Wenn Sie die Shift-Taste Ihrer Tastatur drücken und gedrückt halten und anschließend den Button „Baumstumpf setzen“ der Toolbar anklicken, haben Sie die Möglichkeit, mehrere Baumstümpfe im Territorium zu platzieren, ohne vorher jedes Mal erneut den Button anklicken zu müssen. Solange Sie die Shift-Taste gedrückt halten und eine Kachel anklicken, wird dort ein Baumstumpf platziert.

Pilze werden ähnlich wie Baumstümpfe auf Kacheln platziert. Nutzen Sie dazu den „Pilz setzen“-Button (8. Button von links). Pilze können nicht auf Kacheln platziert werden, auf denen bereits ein anderer Pilz existiert oder auf der sich der Marienkäfer befindet.

Analog zu Baumstümpfen und Pilzen werden Kleeblätter im Territorium platziert, und zwar mit dem „Kleeblatt setzen“-Button (9. Button links). Kleeblätter können nicht auf Kacheln platziert werden, auf denen bereits ein anderes Kleeblatt oder ein Baumstumpf existiert.

Genauso wie der Marienkäfer können auch Baumstümpfe, Pilze und Kleeblätter durch Anklicken und Bewegen der Maus bei gedrückter linker Maustaste im Territorium umplatziert werden.

Mit den Buttons 10 und 11 kann die Weltsicht festgelegt werden. Wenn Sie Button 10 der Toolbar anklicken („Karas Weltsicht“), werden nur Objekte, die Kara sieht, in voller Farbe sichtbar dargestellt. Wenn Sie Button 11 anklicken („globale Weltsicht“) sind alle Objekte des Territoriums voll sichtbar.

Möchten Sie bestimmte Kacheln im Territorium wieder leeren, so aktivieren Sie den „Kachel löschen“-Button (12. Button von links). Klicken Sie anschließend auf die Kacheln, die geleert werden sollen. Der Button ist so lange aktiviert, bis er erneut oder ein anderer Gestaltungsbutton angeklickt wird.

Über die Buttons 13 bis 17 können Sie interaktiv die Befehle move, turnLeft, turnRight, putLeaf und removeLeaf ausführen und damit den Zustand des Marienkäfers und den Zustand des Territoriums verändern.

Über das Menü „Bühne“ können Territorien auch in Datei gespeichert und später wieder geladen werden.

So, jetzt wissen Sie eigentlich alles, was notwendig ist, um das Territorium nach Ihren Wünschen zu gestalten.

5.3 Eingeben eines Scratch-Kara-Programms

Nun begeben wir uns in den Editor-Bereich. Dort werden wir unser erstes Scratch-Kara-Programm schreiben.

Unser erstes Programm soll bewirken, dass der Marienkäfer solange nach vorne läuft, bis er auf einen Baumstumpf oder Pilz trifft.

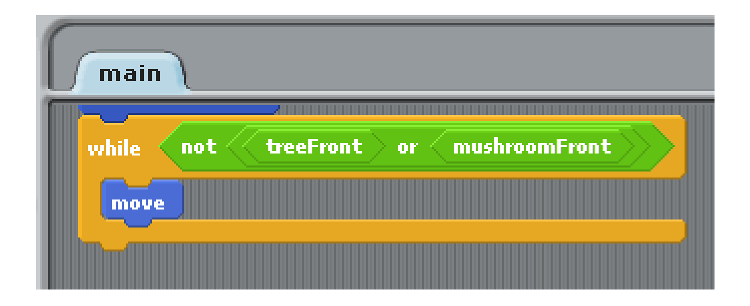


Abb. 5.4: Erstes Scratch-Kara-Programm

Das ist unser erstes Scratch-Kara-Programm.

Im Menü „Programm“ gibt es zwei Menü-Items zum Speichern von Programmen in Dateien und zum wieder Laden von abgespeicherten Programmen. Bei ihrer Aktivierung erscheint eine Dateiauswahl-Dialogbox, in der Sie die entsprechende Auswahl der jeweiligen Datei vornehmen müssen. Achtung: Wenn Sie eine abgespeicherte Datei laden, geht das Programm, das sich aktuell im Editorbereich befindet, verloren (wenn es nicht zuvor in einer Datei abgespeichert wurde).

5.4 Ausführen eines Scratch-Kara-Programms

Nun ist es endlich soweit: Wir können den Marienkäfer bei der Arbeit beobachten. Macht er wirklich das, was wir ihm durch unser Programm beigebracht haben?

Zum Steuern der Programmausführung dienen die Buttons rechts in der Toolbar. Durch Anklicken des „Simulation starten“-Buttons (18. Button von links) starten wir das Programm. Wenn Sie bis hierhin alles richtig gemacht haben, sollte der Marienkäfer loslaufen und wie im Programm beschrieben, den nächsten Baum suchen. Herzlichen Glückwunsch zu Ihrem ersten Scratch-Kara-Programm!

Wollen Sie die Programmausführung anhalten, können Sie dies durch Anklicken des „Simulation pausieren“-Buttons (20. Button von links) erreichen. Der Marienkäfer pausiert so lange, bis Sie wieder den „Simulation starten/fortsetzen“-Button (178. Button von links) anklicken. Dann fährt der Marienkäfer mit seiner Arbeit fort. Das Programm vorzeitig komplett abbrechen, können Sie mit Hilfe des „Simulation beenden“-Buttons (21. Button von links).

Wenn Sie ein Programm mehrmals hintereinander im gleichen Territorium ausführen wollen, können Sie mit dem „Rücksetzen“-Button (22. Button von links) den Zustand des Territoriums wieder herstellen, der vor dem letzten Ausführen des Programms Bestand hatte. Eine komplette Zurücksetzung des Territoriums auf den Zustand beim Öffnen des Scratch-Kara-Simulators ist mit dem „Komplett zurücksetzen“-Button möglich (23. Button von links).

Der Schieberegler ganz rechts in der Menüleiste dient zur Steuerung der Geschwindigkeit der Programmausführung. Je weiter Sie den Knopf nach links verschieben, umso langsamer erledigt der Marienkäfer seine Arbeit. Je weiter Sie ihn nach rechts verschieben, umso schneller flitzt der Marienkäfer durchs Territorium.

Die Bedienelemente zum Steuern der Programmausführung finden Sie übrigens auch im Menü „Simulation“.

5.5 Debuggen eines Scratch-Kara-Programms

„Debuggen eines Programms“ eines Programms bedeutet, dass Sie bei der Ausführung eines Programms zusätzliche Möglichkeiten zur Steuerung besitzen und sich den Zustand des Programms (welcher Befehl wird gerade ausgeführt) in bestimmten Situationen anzeigen lassen können. Das ist ganz nützlich, wenn Ihr Programm nicht das tut, was es soll, und sie herausfinden wollen, woran der Fehler liegt.

Klicken Sie zum Debuggen den „Ablaufverfolgung aktivieren“-Button in der Toolbar an (24. Button von links. Wenn Sie nun Ihr Programm ausführen, wird im Editor rot markiert derjenige Block bzw. Befehl gekennzeichnet, der als nächstes ausgeführt wird. Außerdem können Sie über den „Schritt“-Button (19. Toolbar-Button von links) jeden Befehl einzeln ausführen. Bei einem Prozedur- bzw. Funktionsaufruf wird dabei auch automatisch in die entsprechende Prozedur bzw. Funktion verzweigt.

Sie können zunächst auch einfach das Programm durch Anklicken des „Starten“-Buttons starten und beobachten. Wenn Sie dann den „Pause“-Button drücken, haben Sie anschließend ebenfalls die Möglichkeit der schrittweisen Ausführung ab der aktuellen Position.

Die Ablaufverfolgung kann übrigens jederzeit durch erneutes Klicken des „Ablaufverfolgung“-Buttons deaktiviert bzw. reaktiviert werden.

5.6 Zusammenfassung

Herzlichen Glückwunsch! Wenn Sie bis hierhin gekommen sind, haben Sie Ihr erstes Scratch-Kara-Programm erstellt und ausgeführt. Sie sehen, die Bedienung des Scratch-Kara-Simulators ist gar nicht so kompliziert.

Der Scratch-Kara-Simulator bietet jedoch noch weitere Möglichkeiten. Diese können Sie nun durch einfaches Ausprobieren selbst erkunden oder im nächsten Abschnitt im Detail nachlesen.

6 Bedienung des Scratch-Kara-Simulators

Im letzten Abschnitt haben Sie eine kurze Einführung in die Funktionalität des Scratch-Kara-Simulators erhalten. In diesem Abschnitt werden die einzelnen Funktionen des Simulators nun im Detail vorgestellt. Dabei wird sich natürlich einiges auch wiederholen.

Wenn Sie den Scratch-Kara-Simulator starten, öffnen sich ein Fenster mit dem Titel „Kara-Solist-Simulator“. Abbildung 6.1 skizziert die einzelnen Komponenten des Fensters.

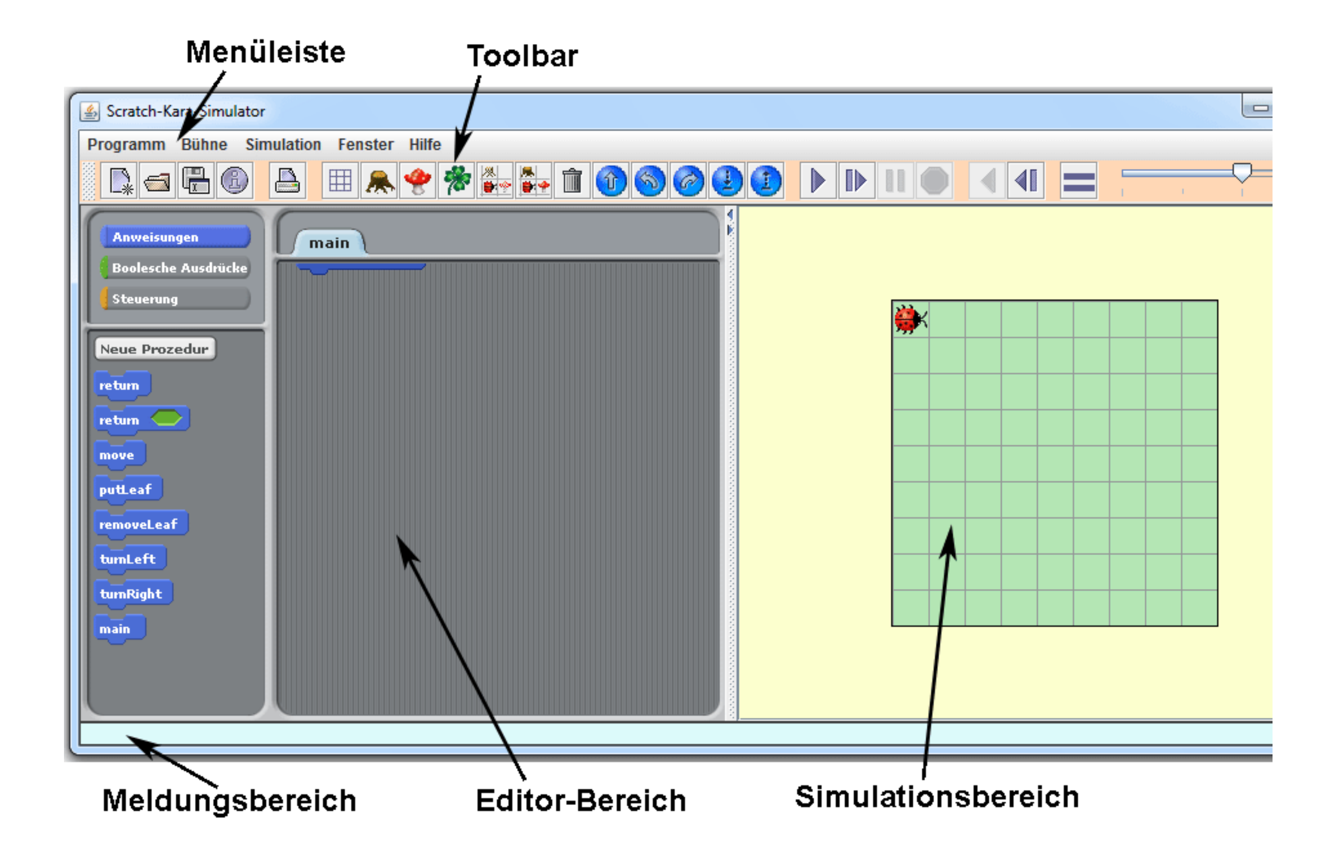


Abb. 6.1: Komponenten des Simulator-Fensters

Die Menüleiste oben im Fenster beinhaltet 5 Menüs. Darunter ist eine Toolbar mit Buttons platziert, über die die wichtigsten Funktionen der Menüs durch Anklicken eines Buttons schneller ausgeführt werden können. Ganz unten im Meldungsbereich werden wichtige Meldungen ausgegeben. Die Eingabe von Scratch-Kara-Programmen erfolgt im Editor-Bereich links und die Ausführung von Scratch-Kara-Programmen wird im Simulationsbereich (auch „Bühne“ genannt) visualisiert.

Als Hauptfunktionsbereiche des Scratch-Kara-Simulators lassen sich identifizieren:

- Verwalten und Editieren von Scratch-Kara-Programmen
- Verwalten und Gestalten von Territorien
- Interaktives Ausführen von Kara-Befehlen
- Ausführen von Scratch-Kara-Programmen
- Debuggen von Scratch-Kara-Programmen

Bevor im Folgenden anhand dieser Funktionsbereiche der Simulator im Detail vorgestellt wird, werden zuvor noch einige Grundfunktionen graphischer Benutzungsoberflächen erläutert.

6.1 Grundfunktionen

In diesem Unterabschnitt werden einige wichtige Grundfunktionalitäten graphischer Benutzungsoberflächen beschrieben. Der Abschnitt ist für diejenigen von Ihnen gedacht, die bisher kaum Erfahrungen mit Computern haben. Diejenigen von Ihnen, die schon längere Zeit einen Computer haben und ihn regelmäßig benutzen, können diesen Abschnitt ruhig überspringen.

6.1.1 Anklicken

Wenn im Folgenden von „Anklicken eines Objektes“ oder „Anklicken eines Objektes mit der Maus“ gesprochen wird, bedeutet das, dass Sie den Maus-Cursor auf dem Bildschirm durch Verschieben der Maus auf dem Tisch über das Objekt platzieren und dann die – im Allgemeinen linke – Maustaste drücken.

6.1.2 Draggen

Wenn im Folgenden von „Draggen eines Objektes“ oder „Verschieben eines Objektes mit der Maus“ gesprochen wird, bedeutet das, dass Sie das entsprechende Objekt mit der Maus anklicken und dann bei gedrückter Maustaste die Maus verschieben. Dabei wird dann automatisch das Objekt auf dem Bildschirm mit verschoben.

6.1.3 Tooltipps

Als *Tooltipps* werden kleine Rechtecke bezeichnet, die automatisch auf dem Bildschirm erscheinen, wenn man den Maus-Cursor auf entsprechende Objekte platziert (siehe Abbildung 6.2). In den Tooltipps werden bestimmte Informationen ausgegeben.



Abb. 6.2: Tooltipp

6.1.4 Button

Buttons sind Objekte der Benutzungsoberfläche, die man anklicken kann und die daraufhin eine bestimmte Aktion auslösen (siehe Abbildung 6.3). Buttons besitzen eine textuelle Beschreibung (z.B. „OK“) oder eine Graphik, die etwas über die Aktion aussagen. Sie erkennen Buttons an der etwas hervorgehobenen Darstellung. Graphik-Buttons sind in der Regel Tooltipps zugeordnet, die die zugeordnete Aktion beschreiben.

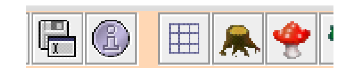


Abb. 6.3: Buttons

6.1.5 Menü

Menüs befinden sich ganz oben in einem Fenster in der so genannten *Menüleiste* (siehe Abbildung 6.4). Sie werden durch einen Text beschrieben (Programm, Bühne, Simulation, ...). Klickt man die Texte an, öffnet sich eine Box mit so genannten *Menü-Items*. Diese bestehen wiederum aus Texten, die man anklicken kann. Durch Anklicken von Menü-Items werden genauso wie bei Buttons Aktionen ausgelöst, die im Allgemeinen durch die Texte beschrieben werden (Speichern, Kopieren, ...). Nach dem Anklicken eines Menü-Items wird die Aktion gestartet und die Box schließt sich automatisch wieder. Klickt man irgendwo außerhalb der Box ins Fenster, schließt sich die Box ebenfalls und es wird keine Aktion ausgelöst.

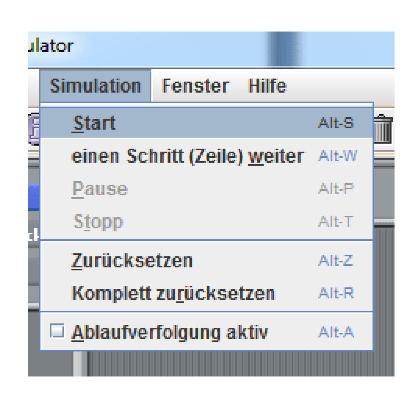


Abb. 6.4: Menüleiste und Menü

Häufig steht hinter den Menü-Items ein weiterer Text, wie z.B. „Strg-O“ oder „Alt-N“. Diese Texte kennzeichnen Tastenkombinationen. Drückt man die entsprechenden Tasten auf der Tastatur, wird dieselbe Aktion ausgelöst, die man auch durch Anklicken des Menü-Items auslösen würde.

Manchmal erscheinen bestimmte Menü-Items etwas heller. Man sagt auch, sie sind ausgegraut. In diesem Fall kann man das Menü-Item nicht anklicken und die zugeordnete Aktion nicht auslösen. Das Programm befindet sich in einem Zustand, in dem die Aktion keinen Sinn machen würde.

6.1.6 Toolbar

Direkt unterhalb der Menüleiste ist die so genannte *Toolbar* angeordnet (siehe Abbildung 6.5). Sie besteht aus einer Menge an Graphik-Buttons, die Alternativen zu den am häufigsten benutzten Menü-Items der Menüs darstellen.



Abb. 6.5: Toolbar

6.1.7 Popup-Menü

Popup-Menüs sind spezielle Menüs, die bestimmten Elementen auf dem Bildschirm zugeordnet sind (siehe Abbildung 6.6). Man öffnet sie dadurch, dass man den Maus-Cursor auf das entsprechende Element verschiebt und danach die rechte Maustaste drückt. Genauso wie bei normalen Menüs erscheint dann eine Box mit Menü-Items.

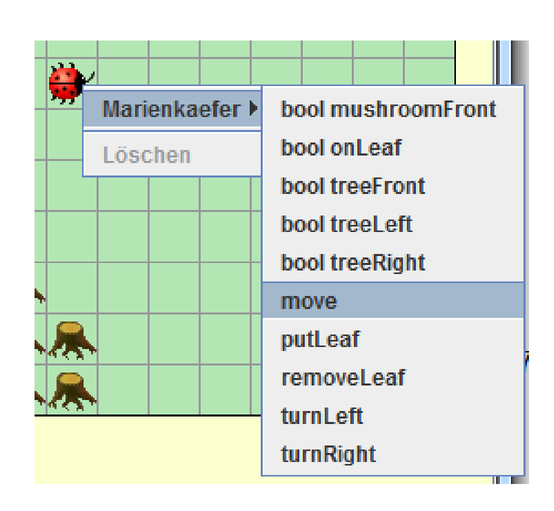


Abb. 6.6: Popup-Menü

6.1.8 Eingabefeld

Eingabefelder dienen zur Eingabe von Zeichen (siehe Abbildung 6.7). Positionieren Sie dazu den Maus-Cursor auf das Eingabefeld und klicken Sie die Maus. Anschließend können Sie über die Tastatur Zeichen eingeben, die im Eingabefeld erscheinen.

6.1.9 Dialogbox

Beim Auslösen bestimmter Aktionen erscheinen so genannte *Dialogboxen* auf dem Bildschirm (siehe Abbildung 6.7). Sie enthalten in der Regel eine Menge von graphischen Objekten, wie textuelle Informationen, Eingabefelder und Buttons. Wenn eine Dialogbox auf dem Bildschirm erscheint, sind alle anderen Fenster des Programms für Texteingaben oder Mausklicks gesperrt. Zum Schließen einer Dialogbox, d.h. um die Dialoxbox wieder vom Bildschirm verschwinden zu lassen, dienen in der Regel eine Menge an Buttons, die unten in der Dialogbox angeordnet sind. Durch Anklicken eines „OK-Buttons“ wird dabei die der Dialogbox zugeordnete Aktion ausgelöst. Durch Anklicken des „Abbrechen-Buttons“ wird eine Dialogbox geschlossen, ohne dass irgendwelche Aktionen ausgelöst werden.

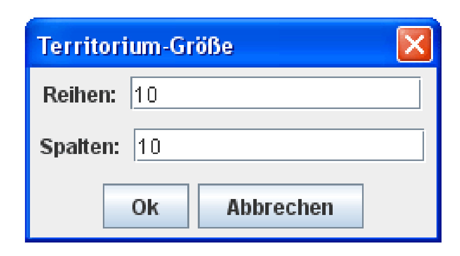


Abb. 6.7: Dialogbox mit Eingabefeldern

6.1.10 Dateiauswahl-Dialogbox

Dateiauswahl-Dialogboxen sind spezielle Dialogboxen, die zum Speichern und Öffnen von Dateien benutzt werden (siehe Abbildung 6.8). Sie spiegeln im Prinzip das Dateisystem wider und enthalten Funktionalitäten zum Verwalten von Dateien und Ordnern.

Im mittleren Bereich einer Dateiauswahl-Dialogbox erscheinen alle Dateien und Unterordner des aktuellen Ordners. Sie sind durch unterschiedliche Symbole repräsentiert. Der eigentliche Zweck von Dateiauswahl-Dialogboxen ist – wie der Name schon sagt – die Auswahl einer Datei. Klickt man auf eine Datei, erscheint der Name automatisch im Eingabefeld „Dateiname“. Dort kann man auch über die

Tastatur einen Dateinamen eingeben. Anschließend wird nach Drücken des OK-Buttons die entsprechende Datei geöffnet bzw. gespeichert.

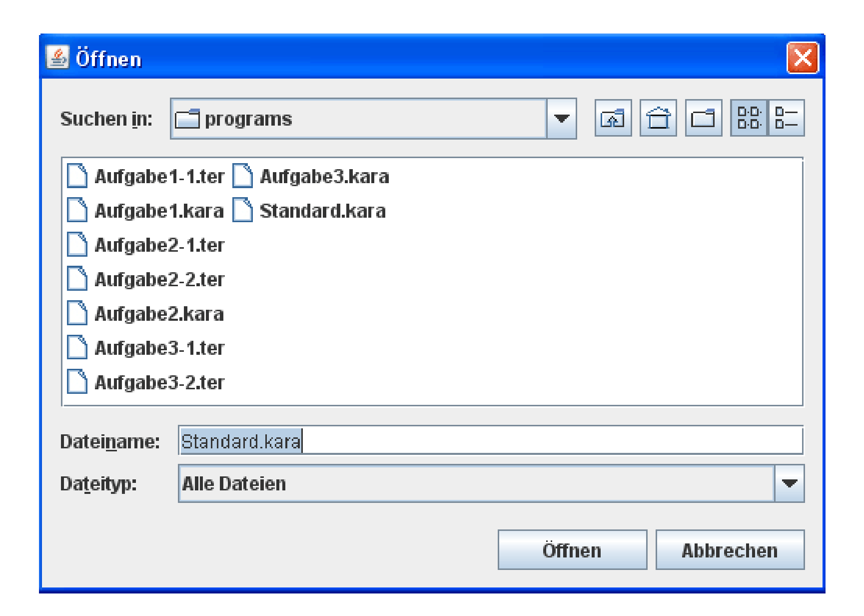


Abb. 6.8: Dateiauswahl-Dialogbox

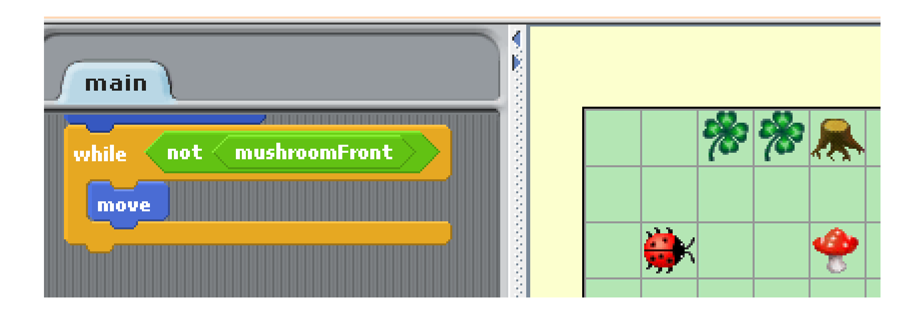
Dateiauswahl-Dialogboxen stellen jedoch noch zusätzliche Funktionalitäten bereit. Durch Doppelklick auf einen Ordner kann man in den entsprechenden Ordner wechseln. Es werden dann anschließend die Dateien und Unterordner dieses Ordners im mittleren Bereich angezeigt. Um zu einem übergeordneten Ordner zurück zu gelangen, bedient man sich des Menüs „Suchen in“, in dem man den entsprechenden Ordner auswählen kann.

Neben dem „Suchen in“-Menü sind noch fünf Graphik-Buttons angeordnet. Durch Anklicken des linken Buttons kommt man im Ordnerbaum eine Ebene höher. Durch Anklicken des zweiten Buttons von links gelangt man zur Wurzel des Ordnerbaumes. Mit dem mittleren Button kann man im aktuellen Ordner einen neuen Unterordner anlegen. Mit den beiden rechten Buttons kann man die Darstellung im mittleren Bereich verändern.

Möchte man einen Ordner oder eine Datei umbenennen, muss man im mittleren Bereich der Dateiauswahl-Dialogbox zweimal – mit Pause zwischendurch – auf den Namen des Ordners oder der Datei klicken. Die textuelle Darstellung des Namens wird dann zu einem Eingabefeld, in der man über die Tastatur den Namen verändern kann.

6.1.11 Split-Pane

Eine Split-Pane ist ein Element, das aus zwei Bereichen und einem Balken besteht. (siehe Abbildung 6.9). Die beiden Bereiche können dabei links und rechts oder oberhalb und unterhalb des Balkens liegen. Wenn Sie den Balken mit der Maus anklicken und bei gedrückter Maustaste nach links oder rechts (bzw. oben oder unten) verschieben, vergrößert sich einer der beiden Bereiche und der andere verkleinert sich. Durch Klicken auf einen der beiden Pfeile auf dem Balken können Sie einen der beiden Bereiche auch ganz verschwinden lassen.



Abbi. 6.9: Split-Pane

6.1.12 Tab-Pane

Ein Tab-Pane besteht unten aus einem größeren Bereich, über den sogenannte Tab-Buttons angeordnet sind (siehe Abbildung 6.10). Jedem Tab-Button sind dabei eigene Inhalte zugeordnet, die im unteren Bereich angezeigt werden, wenn der Tab-Button aktiviert ist. Durch Anklicken der Tab-Buttons kann man zwischen den jeweiligen Inhalten wechseln. Der aktuell aktive Tab-Button wird blau dargestellt. Reicht die Fläche nicht mehr zur Anzeige aller Tab-Buttons wird ganz rechts ein spezieller Tab-Button mit der Beschriftung „…“ angezeigt. Klickt man mit der linken Maustaste auf diesen Button erscheint ein Menü, in dem alle gerade nicht sichtbaren Tab-Buttons angezeigt werden. Beim Klick auf einen der Namen wird der entsprechende Tab-Button ganz links wieder sichtbar und unmittelbar aktiviert.



Abb. 6.10: Tab-Pane

6.2 Verwalten und Editieren von Scratch-Kara-Programmen

Das Erstellen von Programmen bezeichnet man als *Editieren*. Normalerweise wird dabei so genannter Sourcecode in textueller Form geschrieben. Scratch ist jedoch eine visuelle Programmiersprache. Hier bestehen Programme aus visuellen Bausteinen und werden weitgehend mit Hilfe der Maus erstellt. Im Scratch-Kara-Simulator dient der Editor-Bereich zum Editieren von Scratch-Kara-Programmen (siehe Abbildung 6.11).

Der Editor-Bereich ist in drei Unterkomponenten unterteilt:

- Kategorienauswahl
- Blockpalette
- Programmbereich

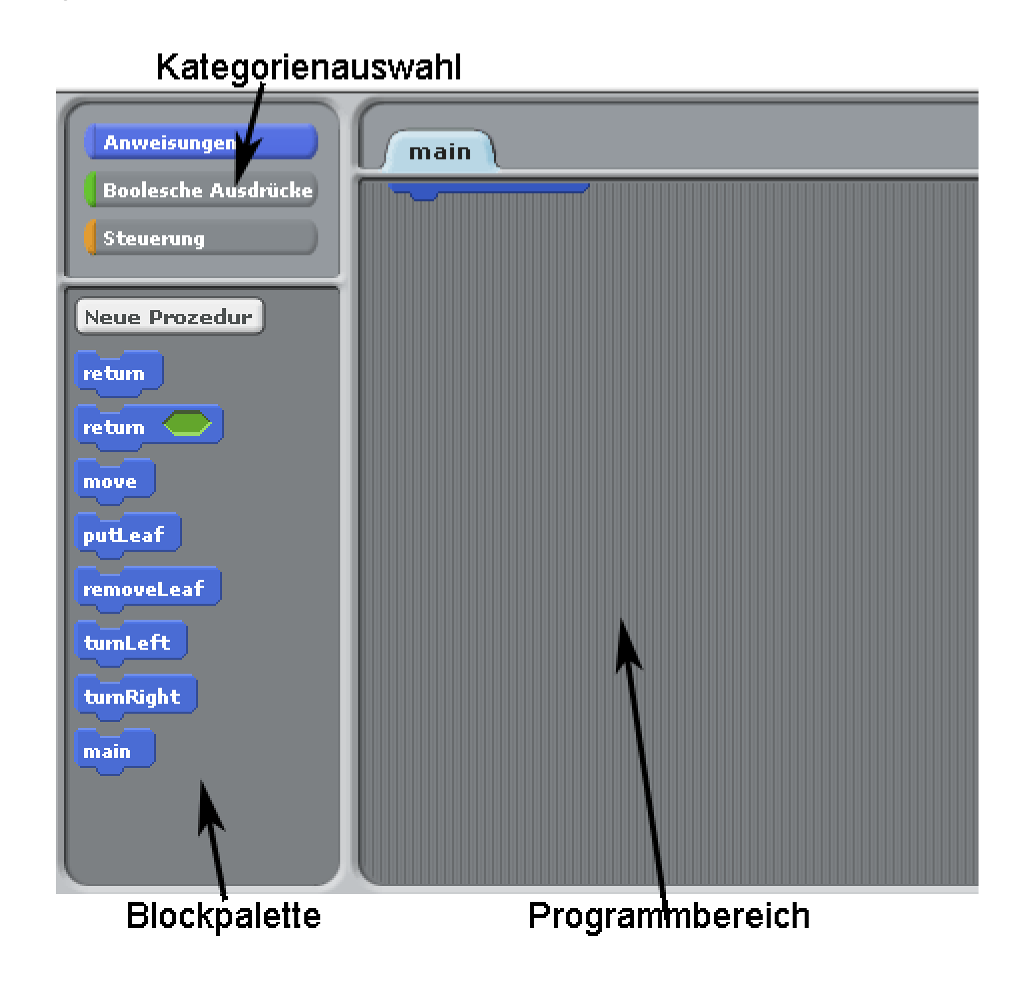


Abb. 6.11: Editor-Bereich des Scratch-Kara-Simulators

In der Kategorienauswahl finden sich die drei Buttons „Anweisungen“, „Boolesche Ausdrücke“ und „Steuerung“. Durch Mausklick auf einen der Buttons erscheinen entsprechende Blöcke der jeweiligen Kategorie in der Blockpalette. Blöcke der Kategorie „Anweisungen“ werden dabei *Anweisungsblöcke*, Blöcke der Kategorie „Boolesche Ausdrücke“ *Boolescher-Ausdruck-Blöcke* und Blöcke der Kategorie „Steuerung“ *Steuerungsblöcke* genannt. Steuerungsblöcke sind spezielle Anweisungsblöcke. Anweisungsblöcke werden blau, Steuerungsblöcke orange und Boolescher-Ausdruck-Blöcke grün dargestellt.

In der Blockpalette werden die Blöcke der in der Kategorienauswahl aktuell ausgewählten Kategorie angezeigt. Blöcke repräsentieren dabei bestimmte Programmbausteine. Blöcke – genauer gesagt Kopien hiervon – lassen sich per Drag-und-Drop mit der Maus in den Programmbereich ziehen, um Programme zu erstellen. Wählen Sie hierzu den entsprechenden Block, klicken Sie ihn mit der Maus (linke Taste) an, ziehen Sie die Maus bei gedrückter linker Taste in den Programmbereich und lassen Sie die Maustaste los.

Im Programmbereich liegen die Scratch-Kara-Programme. Blöcke, die aus der Blockpalette in den Programmbereich gezogen wurden, lassen sich hier per Drag-and-Drop-Aktion weiter hin und herschieben.

Für das Verwalten von Programmen ist das Menü „Programm“ wichtig. Unterhalb der Menüleiste ist eine spezielle Toolbar zu sehen, über die Sie die wichtigsten Funktionen der Menüs auch schneller erreichen und ausführen können. Schieben Sie einfach mal die Maus über die Buttons. Dann erscheint jeweils ein Tooltipp, der die Funktionalität des Buttons anzeigt. Die für das Editieren von Programmen wichtigen Buttons der Toolbar werden in Abbildung 6.12 skizziert.

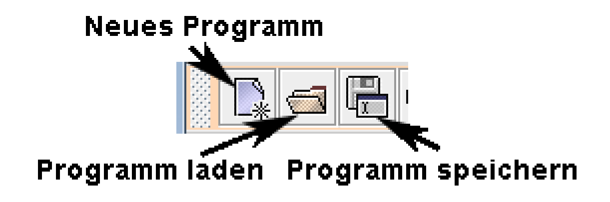


Abb. 6.12: Editor-Buttons der Toolbar

6.2.1 Erstellen eines Scratch-Kara-Programms

Scratch-Kara-Programme bestehen aus visuellen Komponenten, den so genannten Blöcken. Der Umgang mit diesen Blöcken wird in diesem Unterabschnitt beschrieben.

6.2.1.1 Blöcke

Nach dem Start des Scratch-Kara-Simulators können Sie direkt mit dem Erstellen eines Scratch-Kara-Programms loslegen. Ziehen Sie dazu einfach Blöcke aus der Blockpalette mit der Maus in den Programmbereich. Sie können die Blöcke auch jederzeit innerhalb des Programmbereichs verschieben.

6.2.1.2 Stapel

Befindet sich im Programmbereich des Editors ein Anweisungsblock A und zieht man einen anderen Anweisungsblock B in die Nähe (ober- und unterhalb), dann erscheint irgendwann ein transparenter grauer Bereich (siehe Abbildung 6.13). Lässt man B nun los, schnappen Zapfen (unten an den Anweisungsblöcken) und Mulde (oben an den Anweisungsblöcken) der beiden Blöcke ein (man spricht auch von „andocken") und A und B bilden nun eine Anweisungssequenz (in Scratch wird der Begriff „Stapel" verwendet). Zieht man im Folgenden den obersten Block einer Anweisungssequenz im Programmbereich mit der Maus hin und her, wird automatisch der gesamte Block mit verschoben. Eine Anweisungssequenz kann man wieder trennen, indem man einen mittleren Block verschiebt. Darunter liegende Blöcke werden dann mit verschoben, darüber liegende Blöcke bleiben liegen und werden abgetrennt.

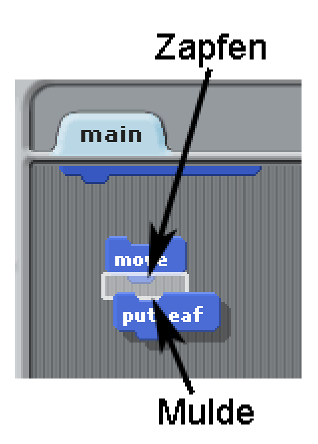


Abbildung 6.13: Bilden von Stapeln

6.2.1.3 Prozedurzapfen

Besondere Zapfen sind die so genannten Prozedurzapfen (siehe Abbildung 6.14). Stapel, die hier angedockt werden, werden ausgeführt, wenn die entsprechende Prozedur aufgerufen wird.

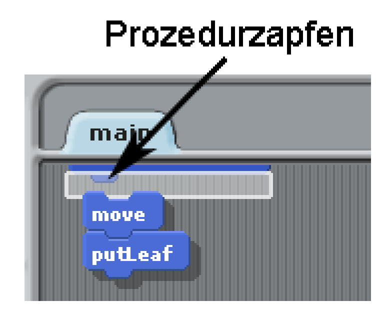


Abbildung 6.14: Prozedurzapfen

6.2.1.4 Verkupplung von Anweisungs- und Boolescher-Ausdruck-Blöcke

Die Steuerungs- und auch bestimmte Anweisungsblöcke enthalten grüne Platzhalter. Hier können Boolesche-Ausdruck-Blöcke angedockt werden.

6.2.1.5 Entfernen von Blöcken

Blöcke, die nicht direkt oder indirekt mit einem Prozedurzapfen verbunden sind, haben, was die Programmausführung betrifft keine Bedeutung. Wieder aus dem Programmbereich entfernt werden können Blöcke bzw. Stapel, indem man sie links aus dem Programmbereich hinauszieht. Alternativ kann man oberhalb eines Blockes im Programmbereich ein Popup-Menü aktivieren und dort das Menü-Item „Entfernen" anklicken. Befindet sich der entsprechende Block in einem Stapel werden auch alle sich darunter befindenden Blöcke entfernt.

6.2.1.6 Prozeduren und Funktionen

Durch Anklicken der beiden Buttons „Neue Prozedur" bzw. „Neue Funktion" lassen sich neue Prozeduren bzw. Funktionen definieren. Es erscheint jeweils ein entsprechender Tab-Button oberhalb des Programmbereichs. Durch Anklicken der Tab-Buttons kann man zwischen einzelnen Prozeduren bzw. Funktionen hin und her wechseln.

Das Schließen von Prozeduren bzw. Funktionen ist möglich, indem man oberhalb des entsprechenden Tab-Buttons ein Popup-Menü aktiviert und hierin das Menü-Item „Schließen" anklickt. Geschlossene Prozeduren bzw. Funktionen lassen sich wieder öffnen, indem man oberhalb des entsprechenden Blocks in der Blockpalette ein Popup-Menü aktiviert und hierin das Menü-Item „Öffnen" anklickt. Eine Alternative zum Öffnen bietet ein Doppelklick mit der Maus auf den entsprechenden Block in der Blockpalette.

Prozeduren bzw. Funktionen können gelöscht oder umbenannt werden, indem man oberhalb des entsprechenden Blocks in der Blockpalette oder oberhalb des entsprechenden Tab-Buttons ein Popup-Menü aktiviert und hierin das Menü-Item „Löschen“ bzw. „Umbenennen“ anklickt. Löschen ist jedoch nur möglich, wenn die Prozedur bzw. Funktion nirgendwo im Programm genutzt wird.

6.2.1.7 Auslagern von Blöcken

Möchte man ganze Stapel oder Teile von Stapeln in Prozeduren bzw. Funktionen auslagern, kann man das erreichen, indem man oberhalb eines Blockes ein Popup-Menü aktiviert und hierin das Menü-Item „Auslagern in Prozedur“ bzw. „Auslagern in Funktion“ anklickt. Es wird dann eine neue Prozedur bzw. Funktion definiert, in die alle Blöcke des Stapels unterhalb des betroffenen Blockes ausgelagert werden. Die ausgelagerten Blöcke werden automatisch durch einen Block der neuen Prozedur bzw. Funktion ersetzt.

Auch Boolescher-Ausdruck-Blöcke lassen sich in Funktionen auslagern. In die neue Funktion wird dann automatisch eine entsprechende Boolesche-Return-Anweisung integriert.

6.2.2 Abspeichern des aktuellen Scratch-Kara-Programms

Normalerweise müssen Sie sich nicht um das Speichern des aktuellen Programms kümmern. Es wird beim Beenden des Scratch-Kara-Simulators automatisch in einer internen Datei abgespeichert. Wenn Sie jedoch ein Programm explizit abspeichern möchten, können Sie in der Toolbar den „Speichern“-Button oder im „Programm“-Menü das „Speichern unter…“-Menü-Item anklicken. Es öffnet sich eine Dateiauswahl-Dialogbox, über die Sie die gewünschte Datei auswählen können.

6.2.3 Öffnen eines einmal abgespeicherten Scratch-Kara-Programms

Möchten Sie ein einmal abgespeichertes Scratch-Kara-Programm wieder in den Editorbereich laden, können Sie dies über den Toolbar-Button „Laden“ oder das Menü-Item „Laden…“ des „Programm“-Menüs tun. Nach dem Anklicken des Items erscheint eine Dateiauswahl-Dialogbox, über die Sie die gewünschte Datei auswählen können.

Achtung: Beim Laden einer abgespeicherten Datei geht der aktuelle Inhalt des Editor-Bereichs verloren! Sie müssen ihn also gegebenenfalls vorher in einer Datei abspeichern.

6.2.4 Erstellen eines neuen Scratch-Kara-Programms

Möchten Sie ein komplett neues Scratch-Kara-Programm erstellen, können Sie den Toolbar-Button „Neu“ oder im „Programm-Menü“ das Menü-Item „Neu“ anklicken. Der aktuelle Programmbereich wird dann geleert.

Achtung: Beim Erstellen eines neuen Scratch-Kara-Programms geht der aktuelle Inhalt des Editor-Bereichs verloren! Sie müssen ihn also gegebenenfalls vorher in einer Datei abspeichern.

6.2.5 Drucken eines Scratch-Kara-Programms

Möchten Sie ein Scratch-Kara-Programm ausdrucken, können Sie den Toolbar-Button „Drucken“ (5. Button von links) oder im „Programm-Menü“ das Menü-Item „Drucken“ anklicken. Weiterhin ist es möglich, Scratch-Programme als png- oder gif-Bilder abzuspeichern (Menü „Programm“, Menü-Item „Als Bild speichern“).

6.3 Verwalten und Gestalten von Territorien

Das Territorium umfasst standardmäßig 9 Reihen und 9 Spalten. Der Marienkäfer steht auf der Kachel ganz oben links, also der Kachel mit den Koordinaten (0/0). Er schaut nach Osten.

In der Toolbar dienen die Buttons 6 – 17 von links zum Gestalten des Territoriums. Über das Menü „Bühne“ können Territorien verwaltet werden (siehe auch Abbildung 6.14).

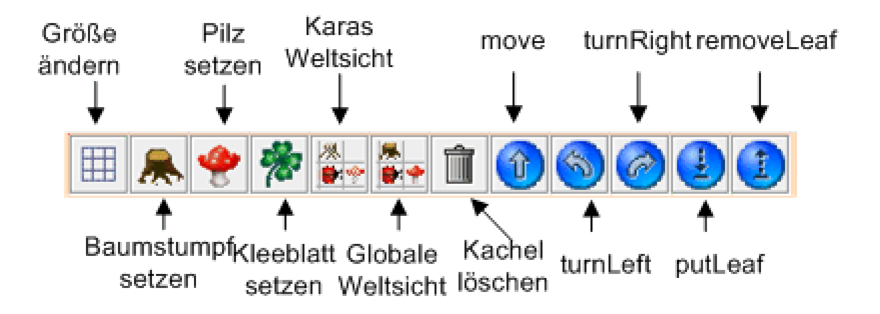


Abb. 6.14: Territorium-Buttons der Toolbar

Normalerweise sollten Sie ein Territorium vor der Ausführung eines Programms gestalten. Es ist jedoch auch möglich, während der Programmausführung noch Umgestaltungen vorzunehmen.

6.3.1 Verändern der Größe des Territoriums

Durch Anklicken des „Größe ändern“-Buttons (6. Toolbar-Button von links) können Sie die Größe des Territoriums verändern. Es öffnet sich eine Dialogbox mit zwei Eingabefelder, in denen Sie die gewünschte Reihen- und Spaltenanzahl eingeben können. Nach Drücken des OK-Buttons schließt sich die Dialogbox und das Territorium erscheint in der angegebenen Größe.

Achtung: Wenn Sie das Territorium verkleinern, werden Objekte, die sich dann außerhalb des Territoriums befinden würden, gelöscht. Steht der Marienkäfer beim Verkleinern auf einer Kachel außerhalb der neuen Territoriumsgröße, wird er auf die Kachel (0/0) versetzt (sich u.U. dort befindende Baumstümpfe oder Pilze werden zuvor gelöscht).

6.3.2 Umplatzieren des Marienkäfers im Territorium

Um den Marienkäfer im Territorium auf eine andere Kachel zu platzieren, klicken Sie ihn mit der Maus an und ziehen ihn bei gedrückter Maustaste auf die gewünschte Kachel.

6.3.3 Setzen der Blickrichtung des Marienkäfers

Um die Blickrichtung des Marienkäfers zu ändern, klicken Sie bitte den „turnLeft“-Button (14. Toolbar-Button von links) an. Bei jedem Klick auf diesen Button dreht sich der Marienkäfer um 90 Grad linksum. Analog funktioniert der „turnRight“-Button (15. Button der Toolbar von links) für eine Rechtsdrehung um 90 Grad.

6.3.4 Platzieren von Baumstümpfen auf Kacheln des Territorium

Um auf einer Kachel des Territoriums einen Baumstumpf zu platzieren, klicken Sie in der Toolbar den „Baumstumpf setzen“-Button (7. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Der Baumstumpf wird dort platziert (insofern sich dort nicht ein anderes Objekt befindet).

Um nicht für jedes Platzieren eines Baumstumpfes erneut den „Baumstumpf setzen“-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Baumstumpf setzen“-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort einen Baumstumpf platzieren.

Baumstümpfe können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie den dortigen Baumstumpf bei gedrückter Maustaste auf die gewünschte Kachel.

6.3.5 Platzieren von Pilzen auf Kacheln des Territorium

Um auf einer Kachel des Territoriums einen Pilz zu platzieren, klicken Sie in der Toolbar den „Pilz setzen"-Button (8. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Der Pilz wird dort platziert (insofern sich dort nicht ein anderer Pilz, ein Baumstumpf oder der Marienkäfer befindet).

Um nicht für jedes Platzieren eines Pilz erneut den „Pilz setzen"-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Pilz setzen"-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort einen Pilz platzieren.

Pilze können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie den dortigen Pilz bei gedrückter Maustaste auf die gewünschte Kachel.

6.3.6 Platzieren von Kleeblättern auf Kacheln des Territorium

Um auf einer Kachel des Territoriums ein Kleeblatt zu platzieren, klicken Sie in der Toolbar den „Kleeblatt setzen"-Button (9. Toolbar-Button von links). Klicken Sie anschließend die entsprechende Kachel an. Das Kleeblatt wird dort platziert (insofern sich dort kein anderes Kleeblatt oder ein Baumstumpf befindet).

Um nicht für jedes Platzieren eines Kleeblatts erneut den „Kleeblatt setzen"-Button anklicken zu müssen, können Sie auch folgendes tun: Drücken Sie die Shift-Taste der Tastatur und Klicken bei gedrückter Shift-Taste den „Kleeblatt setzen"-Button an. Solange Sie nun die Shift-Taste gedrückt halten, können Sie durch Anklicken einer Kachel des Territoriums dort ein Kleeblatt platzieren.

Kleeblätter können auch im Territorium umplaziert werden. Klicken Sie dazu die entsprechende Kachel an und ziehen Sie das dortige Kleeblatt bei gedrückter Maustaste auf die gewünschte Kachel.

6.3.7 Löschen von Kacheln des Territorium

Um einzelne Kacheln des Territoriums zu löschen, d.h. gegebenenfalls vorhandene Pilze, Kleeblätter bzw. Baumstümpfe zu entfernen, müssen Sie zunächst in der

Toolbar den „Kachel löschen“-Button (12. Toolbar-Button von links) anklicken. Dadurch aktivieren Sie die Kachel-Löschen-Funktion. Sie erkennen dies daran, dass der Hintergrund des Buttons nun dunkler erscheint. Solange die Funktion aktiviert ist, können Sie nun durch Anklicken einer Kachel diese Kachel löschen.

Eine Deaktivierung der Kachel-Löschen-Funktion ist durch erneutes Anklicken des „Kachel löschen“-Buttons möglich. Eine Deaktivierung erfolgt automatisch, wenn ein anderer der Buttons zum Territorium-Gestalten gedrückt wird.

Eine weitere Möglichkeit, Kleeblätter, Pilze oder Baumstümpfe im Territorium wieder zu löschen, besteht darin, über den entsprechenden Elementen mit Hilfe der rechten Maustaste ein Popup-Menü zu aktivieren und das darin erscheinende Item „Löschen“ anzuklicken.

6.3.8 Ändern der Weltsicht

Mit den Buttons 10 und 11 kann die Weltsicht festgelegt werden. Wenn Sie Button 10 der Toolbar anklicken („Karas Weltsicht“), werden nur Objekte, die Kara sieht, in voller Farbe sichtbar dargestellt. Andere Objekte erscheinen transparent. Wenn Sie Button 11 anklicken („globale Weltsicht“) sind alle Objekte des Territoriums voll sichtbar.

6.3.9 Kara-Befehle

Über die Buttons 13 bis 17 können Sie interaktiv die Befehle move, turnLeft, turnRight, putLeaf und removeLeaf ausführen und damit den Zustand des Marienkäfers und den Zustand des Territoriums verändern.

6.3.10 Abspeichern eines Territoriums

Sie können einmal gestaltete Territorien in einer Datei abspeichern und später wieder laden. Zum Abspeichern des aktuellen Territoriums aktivieren Sie im Menü „Bühne“ das Menü-Item „Speichern unter…“. Es öffnet sich eine Dateiauswahl-Dialogbox. Hierin können Sie den Ordner auswählen und den Namen einer Datei eingeben, in die das aktuelle Territorium gespeichert werden soll.

Weiterhin ist es möglich, das aktuell Territorium als Bild (gif- oder png-Datei) abzuspeichern. Eine entsprechende Funktion findet sich im „Bühne“-Menü.

6.3.11 Wiederherstellen eines abgespeicherten Territoriums

Abgespeicherte Territorien können mit dem „Laden"-Menü-Item des „Bühne"-Menüs wieder geladen werden. Es erscheint eine Dateiauswahl-Dialogbox, in der Sie die zu ladende Datei auswählen können. Nach dem Anklicken des OK-Buttons schließt sich die Dialogbox und das entsprechende Territorium ist wiederhergestellt.

Achtung: Der Zustand des Territoriums, der vor dem Ausführen der Territorium-Laden-Funktion Gültigkeit hatte, geht dabei verloren. Speichern Sie ihn daher gegebenenfalls vorher ab.

6.4 Interaktives Ausführen von Kara-Befehlen

Sie können einzelne Kara-Befehle oder selbst definierte Funktionen und Prozeduren bspw. zu Testzwecken auch interaktiv ausführen. Aktivieren Sie dazu im Menü „Fenster" den Eintrag „Befehlsfenster". Es öffnet sich das so genannte Befehlsfenster (siehe Abbildung 6.15).

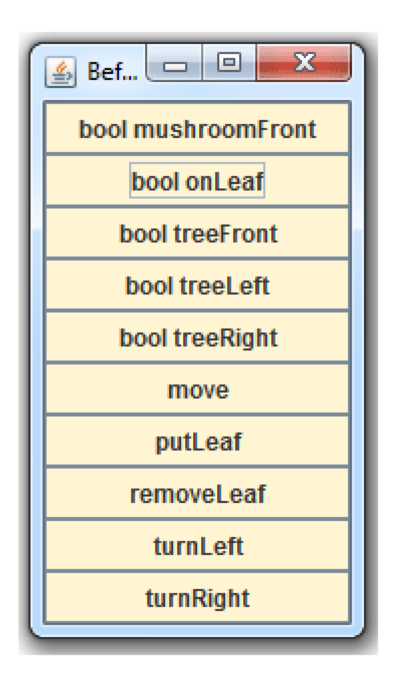


Abb. 6.15: Befehlsfenster

6.4.1 Befehlsfenster

Im Befehlsfenster werden alle Kara-Befehle dargestellt. Beim Anklicken mit der Maus werden die entsprechenden Befehle jeweils ausgeführt.

Bei der Ausführung von Funktionen wird der jeweils gelieferte Wert in einem Dialogfenster dargestellt.

6.4.2 Befehls-Popup-Menü

Alternativ zur Benutzung des Befehlsfensters ist es auch möglich, über ein Popup-Menü die Kara-Befehle interaktiv auszuführen. Sie können dieses Popup-Menü durch Drücken der rechten Maustaste oberhalb des Marienkäfers im Territorium aktivieren (siehe Abbildung 6.16).

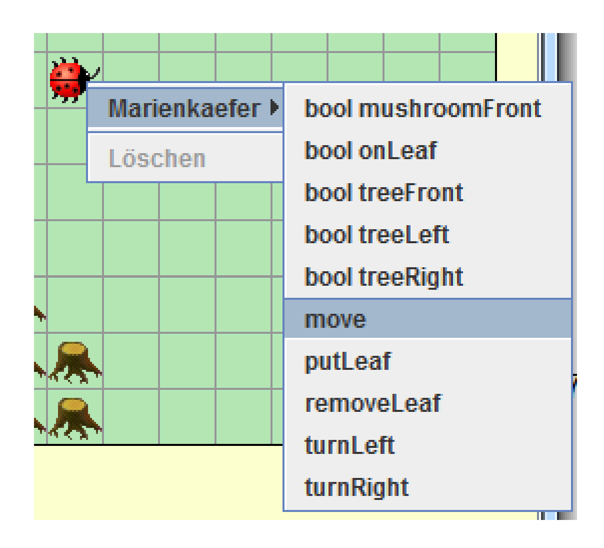


Abb. 6.16: Befehls-Popup-Menü

6.5 Ausführen von Scratch-Kara-Programmen

Ausgeführt werden können Scratch-Kara-Programme mit Hilfe der in Abbildung 6.17 skizzierten Buttons der Toolbar. Alle Funktionen sind darüber hinaus auch über das Menü „Simulation“ aufrufbar.

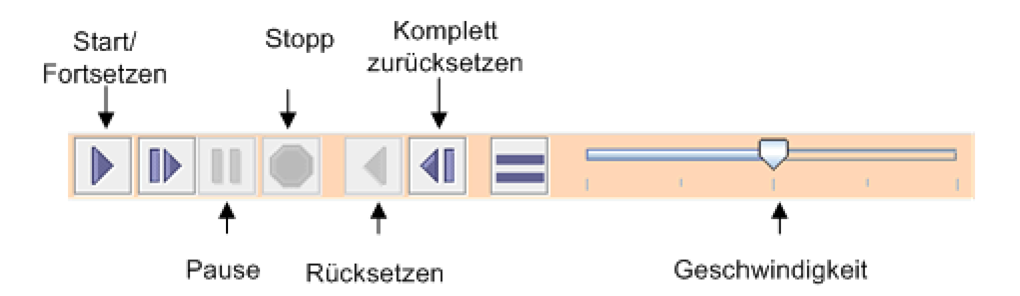


Abb. 6.17: Simulationsbuttons der Toolbar

6.5.1 Starten eines Scratch-Kara-Programms

Bevor ein Scratch-Kara-Programm ausgeführt werden kann, muss es erfolgreich compiliert worden sein. Gestartet werden kann das aktuelle Scratch-Kara-Programm dann durch Anklicken des „Start/Fortsetzen“-Buttons (18. Toolbar-Button von links).

Nach dem Starten eines Scratch-Kara-Programms wird der Marienkäfer im Territorium aktiv und tut das, was das Programm ihm vorgibt. Während des Ausführens eines Scratch-Kara-Programms wird der Editor-Bereich ausgegraut, d.h. es können während der Ausführung eines Programms keine Änderungen am Sourcecode durchgeführt werden.

6.5.2 Stoppen eines Scratch-Kara-Programms

Die Ausführung eines Scratch-Kara-Programms kann durch Anklicken des „Stopp“-Buttons (21. Button der Toolbar von links) jederzeit abgebrochen werden.

6.5.3 Pausieren eines Scratch-Kara-Programms

Möchten Sie ein in Ausführung befindliches Programm (kurzfristig) anhalten, können Sie dies durch Anklicken des „Pause“-Buttons (20. Button der Toolbar von links) tun. Wenn Sie anschließend auf den „Start/Fortsetzen“-Button klicken, wird die Programmausführung fortgesetzt.

6.5.4 Während der Ausführung eines Scratch-Kara-Programms

Treten bei der Ausführung eines Programms Laufzeitfehler auf, z.B. wenn ein Marienkäfer gegen einen Baumstumpf rennt, wird eine Dialogbox geöffnet, die eine entsprechende Fehlermeldung enthält. Nach dem Anklicken des OK-Buttons in der Dialogbox wird das Scratch-Kara-Programm beendet. Weiterhin öffnet sich das Konsolen-Fenster, in dem ebenfalls die Fehlermeldung ausgegeben wird.

Während der Ausführung eines Scratch-Kara-Programms ist es durchaus noch möglich, das Territorium umzugestalten, also bspw. neue Pilze und Kleeblätter zu platzieren.

6.5.5 Einstellen der Geschwindigkeit

Mit dem Schieberegler ganz rechts in der Toolbar können Sie die Geschwindigkeit der Programmausführung beeinflussen. Je weiter links der Regler steht, desto langsamer wird das Programm ausgeführt. Je weiter Sie den Regler nach rechts verschieben, umso schneller flitzt der Marienkäfer durchs Territorium.

6.5.6 Wiederherstellen eines Territoriums

Beim Testen eines Programms recht hilfreich ist der „Rücksetzen"-Button (22. Button der Toolbar von links). Sein Anklicken bewirkt, dass das Territorium in den Zustand zurückversetzt wird, den es vor dem letztmaligen Start eines Programms inne hatte.

Über den „Komplett Zurücksetzen"-Button (23. Button der Toolbar von links) ist eine Rücksetzung des Territoriums in den Zustand möglich, der beim Öffnen des Scratch-Kara-Simulators gültig war. Sollte es irgendwann einmal bei der Benutzung des Scratch-Kara-Simulators zu unerklärlichen Fehlern können, ist es mit Hilfe dieses Buttons möglich, den Scratch-Kara-Simulator zu reinitialisieren.

6.6 Debuggen von Scratch-Kara-Programmen

Debugger sind Hilfsmittel zum Testen von Programmen. Sie erlauben es, während der Programmausführung den Zustand des Programms zu beobachten und gegebenenfalls sogar interaktiv zu ändern. Damit sind Debugger sehr hilfreich, wenn es um das Entdecken von Laufzeitfehlern und logischen Programmfehlern geht.

Der Debugger des Scratch-Kara-Simulators ermöglicht während der Ausführung eines Scratch-Kara-Programms das Beobachten des Programmzustands. Sie können sich während der Ausführung eines Scratch-Kara-Programms anzeigen lassen, welche Anweisung des Sourcecodes gerade ausgeführt wird,

Die Funktionen des Debuggers sind eng mit den Funktionen zur Programmausführung verknüpft. Sie finden die Funktionen im Menü „Simulation". Es bietet sich jedoch an, die entsprechenden Buttons der Toolbar zu verwenden. Neben dem „Start/Fortsetzen"-, dem „Pause"- und dem „Stopp"-Button gehören die zwei Buttons „Schrittweise Ausführung" und „Ablaufverfolgung" zu den Debugger-Funktionen (siehe auch Abbildung 6.18).

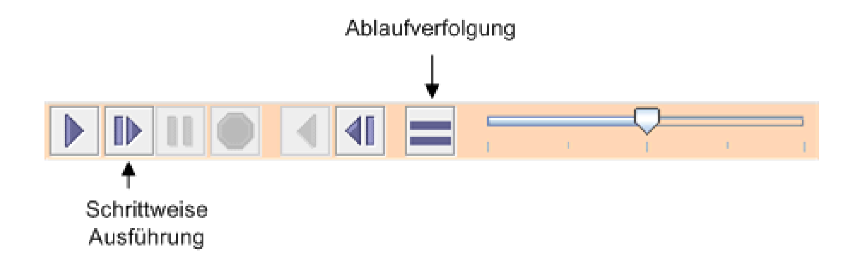


Abb. 6.18: Debugging-Buttons der Toolbar

6.6.1 Beobachten der Programmausführung

Durch Anklicken des Buttons „Ablaufverfolgung“ (24. Button der Toolbar von links) aktivieren bzw. (bei erneuten Anklicken) deaktivieren Sie die Ablaufverfolgung des Debuggers.

Ist die Ablaufverfolgung aktiviert, wird der Block des Programms, der als nächstes ausgeführt wird, rot markiert. Bei einem Prozedur- bzw. Funktionsaufruf wird in die entsprechende Prozedur bzw. Funktion gesprungen und entsprechende Tab-Buttons unter Umständen automatisch geöffnet.

Auch während die Ablaufverfolgung aktiviert ist, können Sie die Programmausführung durch Anklicken des „Pause“-Buttons anhalten und durch anschließendes Anklicken des „Start/Fortsetzen“-Buttons wieder fortfahren lassen. Auch die Geschwindigkeit der Programmausführung lässt sich mit dem Schieberegler anpassen.

6.6.2 Schrittweise Programmausführung

Mit dem Toolbar-Button „Schrittweise Ausführung“ (13. Button der Toolbar von links) ist es möglich, ein Programm schrittweise, d.h. Anweisung für Anweisung auszuführen. Immer, wenn Sie den Button anklicken, wird die nächste Anweisung (bzw. Zeile) ausgeführt.

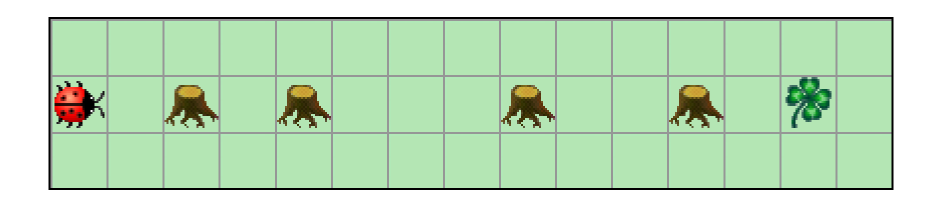
Sie können das Programm mit der schrittweisen Ausführung starten. In diesem Fall wird automatisch die Ablaufverfolgung eingeschaltet. Sie können jedoch auch zunächst das Programm normal starten, dann pausieren und ab der aktuellen Anweisung schrittweise ausführen. Eine normale Programmweiterführung ist jederzeit durch Klicken des „Start“-Buttons möglich.

7 Beispielprogramme

Dem Scratch-Kara-Simulator sind drei Beispielprogramme beigefügt, die Sie über das Menü „Programm“ laden können (Menü-Item „Laden…“). Weitere Aufgaben inkl. Musterlösungen finden Sie bspw. auf der Website von Kara.

7.1 *Kleeblatt suchen*

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara soll ein Kleeblatt finden, das sich in der gleichen Zeile (oder Spalte) befindet wie er selbst. Zwischen ihm und dem Kleeblatt können Bäume stehen, wobei nie zwei Bäume direkt nebeneinander stehen.

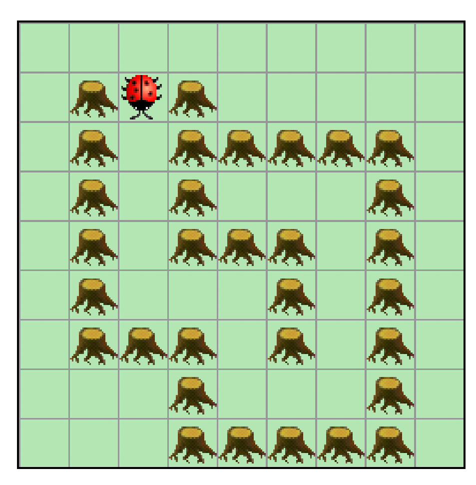


Lösung: Aufgabe1.src

Beispiel-Territorien: Aufgabe1-1.ter und Aufgabe1-2.ter

7.2 *Labyrinth*

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara sucht das Ende eines einfachen Labyrinths bestehend aus Bäumen, wobei keine Löcher in den Baumreihen auftreten. Das Ende des Labyrinths ist eine "Sackgasse".

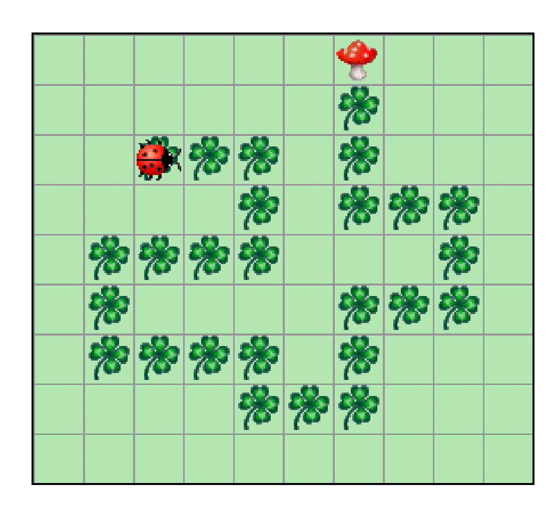


Lösung: Aufgabe2.src

Beispiel-Territorien: Aufgabe2-1.ter und Aufgabe2-2.ter

7.3 Pacman

Aufgabe (von http://www.gierhardt.de/informatik/info11/javakara.html): Kara spielt Pacman: Er steht auf dem ersten Kleeblatt einer langen Spur von Kleeblättern, die vor einem Pilz endet. Er soll alle Kleeblätter auffressen.



Lösung: Aufgabe3.scr

Beispiel-Territorien: Aufgabe3-1.ter und Aufgabe3-2.ter

8 Literatur zum Erlernen der Programmierung

Der Scratch-Kara-Simulator ist ein Werkzeug, mit dem Programmieranfänger einen ersten Eindruck davon vermittelt bekommen können, um was es bei der Programmierung geht.

Wenn Sie diese erste Hürde erfolgreich gemeistert haben, empfehle ich Ihnen zunächst auf den Java-Kara-Simulator und die Programmierung mit der textuellen Programmiersprache Java umzusteigen.

Zum Kara-Modell und dem Java-Kara-Simulator gibt es viele Informationen und Materialien auf der Kara-Website: http://www.swisseduc.ch/informatik/karatojava/ Trotzdem rate ich dringend dazu, beim Erlernen der Programmierung auch Lehrbücher zu nutzen.

Es gibt heutzutage unzählige Lehrbücher zu Java und es ist schwer zu sagen, für wen welches Buch das geeignetste ist. Viele Bücher stehen bei Amazon oder Google-Books zumindest auszugsweise online zur Verfügung und ich kann nur empfehlen, über diese Online-Angebote selbst einmal in die Bücher hineinzuschnuppern. Die drei folgenden Bücher kann ich insbesondere für Programmieranfänger empfehlen:

- Boles, D.: „Programmieren spielend gelernt mit dem Java-Hamster-Modell", Vieweg+Teubner.
- Ratz, D., Scheffler, J., Seese, D. und Wiesenberger, J.: „Grundkurs Programmieren in Java", Hanser-Verlag.
- Heinisch, C., Müller, F. und Goll, J..: „Java als erste Programmiersprache", Vieweg+Teubner.